UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

AMBIENTE DE PROGRAMAÇÃO MULTIMODAL ACESSÍVEL PARA CRIANÇAS COM DEFICIÊNCIA VISUAL

Gonçalo Duarte Cipriano Cardoso

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Engenharia de Software

Dissertação orientada por:
Prof. Doutor Tiago João Vieira Guerreiro
Doutora Ana Cristina de Oliveira Tomé Lopes Pires

2020

Agradecimentos

Aos meus pais, família e namorada agradeço pelo apoio durante toda a minha vida académica. Ao orientador Prof. Tiago Guerreiro e co-orientadora Drª.Ana Pires agradeço pela disponibilidade, acompanhamento e partilha de conhecimentos durante a dissertação. Um especial agradecimento à Lúcia Abreu, pelas ideias partilhadas e pela forte contribuição ao desenhar e imprimir blocos tangíveis utilizados neste projeto. Agradeço à FCUL e ao LASIGE por proporcionarem as condições necessárias, durante realização do projeto final.

Resumo

Atualmente, a tecnologia encontra-se cada vez mais presente no nosso quotidiano, sendo que alguns dispositivos são indispensáveis, uma vez que permitem a utilização de ferramentas que nos ajudam na nossa organização pessoal, social e profissional.

Para o desenvolvimento de dispositivos e ferramentas, são necessários conhecimentos de programação que são utilizados na produção de algoritmos e na criação de sistemas.

A aprendizagem de conceitos de programação não é fácil, tornando-se ainda mais difícil para pessoas, em que o acesso a estes conteúdos é mais limitado, nomeadamente a pessoas cegas ou com baixa visão. Existe, por este motivo, uma grande necessidade de tornar a sua aprendizagem mais simples e acessível. Estudos, efetuados no passado revelam que a aquisição destes conhecimentos em idades muito precoces possibilita a sua compreensão de uma forma mais eficaz.

Uma das contribuições deste trabalho foi desenvolver um sistema que permite a aprendizagem de conceitos de programação a crianças com deficiências visuais, através da utilização de tangíveis ou do reconhecimento de voz, para que um robô se movimente num mapa construído com *Legos*.

O sistema possibilita que sejam construídos caminhos com peças de diferentes cores, permitindo o seu mapeamento por uma aplicação que pode ser executada num *tablet* ou *smartphone*. Esta tem a capacidade de obter sequências através de blocos tangíveis em que são colocadas peças, que contêm relevos permitindo a identificação táctil por crianças cegas, ou através de reconhecimento por voz, no qual a aplicação faz várias questões sobre o que o utilizador pretende fazer, sendo que este ao responder vai construindo o algoritmo. O robô de pequenas dimensões executa a sequência efetuada atravessando o caminho construído.

Palavras-chave: Conceitos de programação, Tangíveis, Reconhecimento de voz, Robô, Caminho de *Legos*

Abstract

Currently, technology is increasingly present in our daily lives, and some devices are indispensable, since they allow the use of tools that help us in our personal, social and professional organization.

For the creation of devices and tools, knowledge of programming is required, for the production of algorithms that lead to the creation of these.

Learning programming concepts is not easy, making it even more difficult for people, where access to these contents is more complicated, such as blind and low vision people. For this reason, there is a great need to make learning more simple and accessible for people with visual impairments. Some studies carried out show that the acquisition of this knowledge at very early ages makes it possible to understand it more effectively.

One of the contributions of this work was to develop a system, which allows the learning of programming concepts to children with visual impairments through the use of tangibles or voice recognition so that a robot moves on a map built with *Legos*.

The system allows paths to be built with pieces of different colors, allowing their recognition by an application that can be run on a *tablet* or *smartphone*. It also allows obtaining sequences through tangible blocks in which pieces containing reliefs are placed allowing tactile identification by blind children, or through voice recognition, in which the application asks several questions about what the user can do and that the user responding builds the algorithm. The small robot executes the sequence carried out across the constructed path.

Keywords: Programming concepts, Tangibles, Speech recognition, Robot, Path of *Legos*

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O pensamento computacional encontra-se cada vez mais difundido nas escolas, sendo lecionado em disciplinas como as TIC (Tecnologias da Informação e da Comunicação) [5] em que são atualmente realizadas atividades de programação. Este permite um desenvolvimento cognitivo e mental das crianças que vai muito além dos conceitos aprendidos, em sala de aula, desenvolvendo capacidades que têm influência no seu quotidiano e a longo prazo, nas suas vidas pessoais e profissionais [50].

Todos os sistemas de software são desenvolvidos e criados por meio de linguagens de programação, que são compostas por diversos conceitos, necessários para a sua aplicação. Quando usados, podemos obter uma sequência de instruções que permite a resolução de um determinado problema, ao qual chamamos de algoritmo.

Com a necessidade de aprender programação, começaram a surgir sistemas que possibilitam o ensino desses conceitos nas salas de aulas, simulando a construção de código através de blocos virtuais, que quando encaixados formam sequências de instruções [35, 20]. Contudo, a maioria destes sistemas tem limitações, devido ao facto de não se encontrarem adaptados, para a utilização de pessoas com deficiências visuais.

A aprendizagem de conceitos de programação deveria ser acessível a qualquer pessoa, no entanto, verifica-se que isso não é uma realidade, por ser inalcançável a pessoas com limitações, nomeadamente cegas ou com baixa visão [39, 48, 40].

Para colmatar estas desigualdades, atualmente têm surgido alguns sistemas que oferecem soluções para a resolução destes problemas. Na generalidade, estes utilizam blocos tangíveis possibilitando a que pessoas cegas os percecionem e os disponham em sequência, criando os seus algoritmos, com o objetivo de movimentar robôs [44, 47].

Por forma a ultrapassar estas restrições, decidi realizar um sistema, que proporciona o desenvolvimento do pensamento computacional em crianças cegas e com baixa visão. Este sistema permite inovar em diversas vertentes, uma vez que possibilita a sua utilização em contextos diferentes (em sala de aula ou em casa), tornando-se bastante económico na aquisição dos materiais. Composto por dois modos de interação que facilitam a aprendizagem de conceitos de programação, permitindo a criação de sequências através de blocos

tangíveis e do reconhecimento de voz, com o objetivo de movimentar um robô, sobre um mapa construído com *Legos*.

1.1 Motivação

Atualmente, a tecnologia desempenha um papel predominante nas nossas vidas, devido à permanente utilização de diferentes dispositivos, que facilitam tarefas profissionais ou pessoais, no nosso quotidiano. Para as realizar, existe um conjunto de passos que devem ser seguidos, que apenas são possíveis de concretizar, se o pensamento computacional for desenvolvido [40, 51, 22].

A aprendizagem do pensamento computacional tem vindo a demonstrar-se como um forte aliado ao desenvolvimento cognitivo, mental e emocional em crianças com idades precoces, através do ensino de conceitos de programação nas escolas [19, 51, 37, 50]. Existem sistemas que suportam este tipo de ensino em escolas, que consistem na criação de algoritmos através da colocação sequencial de blocos virtuais, em dispositivos como *tablets* ou *smartphones*.

Apesar dos benefícios anteriormente referidos, existem pessoas que têm limitações visuais, principalmente cegas e de baixa visão que não têm a possibilidade de usufruir da aprendizagem de conceitos de programação e, por consequência, não desenvolvem o pensamento computacional [39, 48, 40].

Hoje em dia, já existem alguns sistemas que apresentam soluções que visam atenuar as limitações no acesso à aprendizagem de conceitos de programação. Estes, na sua maioria, possibilitam a criação de algoritmos, através de blocos tangíveis que correspondem a diferentes instruções que são colocadas de forma sequencial, com o objetivo de movimentar um robô que fornece *feedback* sobre as ações que está a executar, percebendo, desta forma, que se efetuaram os movimentos desejados [44, 47].

Assim, de modo a combater as limitações, que existem na aprendizagem de conceitos de programação para pessoas cegas e de baixa visão, defini como objetivo criar um sistema que fornece um ambiente multimodal, possibilitando a criação de sequências através de dois modos diferentes de interação: o modo por blocos tangíveis, em que são colocadas várias peças, que definem uma sequência ou através do modo por voz, no qual existe uma estratégia conversacional, sendo efetuadas questões sobre as instruções que se pretendem executar. Após a construção sequencial, torna-se possível a movimentação de um robô, que se desloca num caminho construído com *legos*.

1.2 Objetivos

O principal objetivo deste projeto é desenhar, desenvolver e avaliar uma solução que permita a crianças com deficiências visuais, programar robôs, fomentando a aprendiza-

gem de conceitos de programação.

Para ser possível atingir o objetivo acima referido, entendeu-se como necessário:

- Desenhar uma solução que acomodasse as diferenças não só visuais, mas também atenuasse as dificuldades cognitivas na aprendizagem da criança, permitindo o desenvolvimento computacional.
- Criar um ambiente de programação que possibilitasse às crianças desenvolver o pensamento computacional, através de um ambiente multimodal.
  - Desenvolver um sistema económico e de fácil montagem.
  - Desenvolver uma solução com blocos tangíveis, de modo a criar sequências que permitam obter um *feedback* tátil.
  - Desenvolver uma solução que utilize o reconhecimento de voz, para que possibilite uma estratégia conversacional, na qual através de um conjunto de questões, se obtenha a sequência.
  - Desenvolver uma solução que permita a execução de um robô e que forneça *feedback* sonoro sobre os seus movimentos.
  - Desenvolver uma solução que possibilite a construção de um caminho arbitrário construído com *Legos*.
- Avaliar a adequabilidade do sistema desenvolvido.

1.3 Ambiente Multimodal Acessível

Para satisfazer os objetivos anteriormente definidos, decidiu-se criar um sistema que vise atenuar as limitações do acesso à aprendizagem de conceitos de programação e dificuldades sentidas ao nível do desenvolvimento do pensamento computacional.

De forma a criar um sistema com estas características, iniciou-se um processo de pesquisa acerca dos protótipos já existentes, identificando funcionalidades e os componentes destes. Assim, foi possível reunir esta informação que se encontra exposta no capítulo 2 (Trabalho Relacionado).

Posteriormente, procedeu-se à definição de funcionalidades que considerámos essenciais e inovadoras, relativamente aos sistemas anteriormente apresentados. Para isso, efetuaram-se protótipos em várias fases e com diferentes graus de detalhe até chegar ao protótipo final, tendo sempre como objetivo, um sistema que fosse acessível a crianças cegas ou de baixa visão, económico, de fácil montagem e utilização em ambientes diferentes (em casa ou sala de aula), encontrando-se processo descrito no capítulo 3 (Desenho do Sistema).

Em seguida, procedeu-se ao desenvolvimento do sistema. Este é composto por vários elementos: um *board* que permite o encaixe de *Legos*, blocos tangíveis e um tripé que suporta um *tablet* que executa uma aplicação.

O sistema é de fácil montagem, pois, após colocar o *tablet* no tripé, encaixar os *legos* que formam o caminho e centrá-lo com o dispositivo, encontra-se pronto a ser utilizado. Este possibilita a aprendizagem de conceitos de programação, por meio de sequências que são criadas através de dois modos: um por blocos que consiste na colocação de peças tangíveis e o outro por voz, no qual através de uma estratégia conversacional, são efetuadas questões acerca das instruções a utilizar. Por último, um robô recebe a sequência e executa os movimentos no caminho construído com *legos*, encontrando-se o processo relatado no capítulo 4 (Implementação).

Durante o desenvolvimento do sistema teve-se em conta os objetivos definidos, podendo dizer-se que foram cumpridos, obtendo no final do projeto um sistema economicamente viável e de fácil montagem, que permite uma interação multimodal, possibilitando a criação de qualquer caminho e a movimentação de um robô.

Capítulo 2

Trabalho Relacionado

Neste capítulo são discutidas diferentes abordagens de sistemas construídos com o objetivo de ensinar conceitos e lógica de programação, promovendo o desenvolvimento computacional em crianças. Estas abordagens distinguem-se entre si pelo uso de diferentes interfaces, tais como: interfaces tangíveis, interfaces gráficas e interfaces por voz.

2.1 Aprendizagem do Pensamento Computacional

Nesta secção, é abordado o pensamento computacional, qual a sua importância e como pode ser ensinado a crianças.

Hoje em dia, é possível verificar que existe um aumento da dependência do número de sistemas tecnológicos e da sua utilização no nosso quotidiano [40, 51, 22] e como tal, a tecnologia desempenha um papel cada vez mais importante, nas mais diversas áreas da sociedade. Para a construção destes sistemas é necessário que exista um desenvolvimento do pensamento computacional, que é caracterizado por criar soluções simples e eficazes para diversos problemas, tirando partido de conceitos de programação.

Existem diferentes perspetivas do que é o pensamento computacional no entanto, a que considero mais relevante é: "O pensamento computacional que é um processo de pensamento que se encontra envolvido na formulação de problemas e das suas soluções, para que estas sejam representadas por um agente de processamento de informações."[50]

O ensino de conceitos de programação a crianças é importante, porque permite o desenvolvimento do pensamento computacional,podendo melhorar o seu raciocínio e a forma como pensam e lidam com os problemas do quotidiano. Estes conceitos podem ser aprendidos através de mecanismos, como a descomposição, uma divisão de um problema em várias partes, que ajuda a diminuir a dificuldade, nos desafios que são propostos e da abstração [50] que é necessária para focar o essencial e não apenas os detalhes, permitindo o reconhecimento de padrões. Finalmente, estes mecanismos traduzem-se num pensamento algorítmico, possibilitando a criação de uma sequência lógica, a partir de um conjunto de instruções que levam à resolução do problema.

2.2 Interfaces Gráficas

Sistemas com interfaces gráficas são aqueles em que normalmente a sua interação ocorre num dispositivo digital, tais como o computador, *tablet* ou *smartphone*, contendo vários elementos visuais a serem executados e visualizados no ecrã destes dispositivos.

As interfaces gráficas que abordamos nesta secção, na sua grande maioria são interfaces que permitem ao utilizador arrastar blocos com cores e formatos diferentes, correspondendo estes a uma instrução específica, que quando encaixados uns nos outros, permitem formar um algoritmo.

Uma das abordagens é o *Scratch* [35] que consiste numa linguagem de programação por blocos, para a qual é fornecido um ambiente de programação muito simples de navegar, através dos seus multipainéis.

O painel da esquerda encontra-se dividido por categorias de instruções que podem ser utilizadas no desenvolvimento de programas. Estas classificam-se em 8 categorias, tais como: *motion blocks*, referentes a todos os blocos relacionados com movimento, *looks blocks*, referentes à aparência, *sound blocks*, que controlam o som, *event blocks*, que, a partir de uma ação, despoletam determinados eventos, *sensing blocks*, que detetam variados aspetos no decorrer do programa, *operators blocks*, que realizam funções matemáticas, *variables blocks* que guardam variáveis e listas de elementos, *control blocks*, que controlam *scripts*, *pen blocks*, que permitem efetuar alterações com uma caneta a elementos de execução. Após a escolha de uma destas categorias, é possível visualizar no painel, todas as suas instruções (ver figura 2.1).

Na construção do programa é necessário realizar um movimento *drag and drop* das instruções para o painel inserido no centro da interface e para percecionar as sequências possíveis de realizar com as diferentes instruções, estas apresentam formas diferentes e sugestivas que facilitam o seu encaixe noutras, agregando-se em blocos de código, criando, assim, o seu programa. Existem algumas instruções que ajudam na sua construção e que têm funções específicas, podendo dividir-se em *command blocks*, *function blocks*, *trigger blocks* e *control structure blocks*.

Os *command blocks* podem criar uma *stack* (pilha), as *function blocks* retornam apenas uma função, os *trigger blocks* são executados sempre que um determinado evento ocorre e os *control structure* permitem que vários blocos se encaixem.

O *Scratch* [35] permite também três *data types*: *Boolean*, *Number* e *String*, possibilitando programas orientados a objetos e concorrência ou *threading*.

Por último, é possível correr o programa que foi realizado e, quando executado, pode ocorrer um erro, nesse caso, a instrução aparece em redor com uma borda vermelha, permitindo ao utilizador fazer *debug* e corrigir o erro que cometeu no seu código (ver figura 2.2).

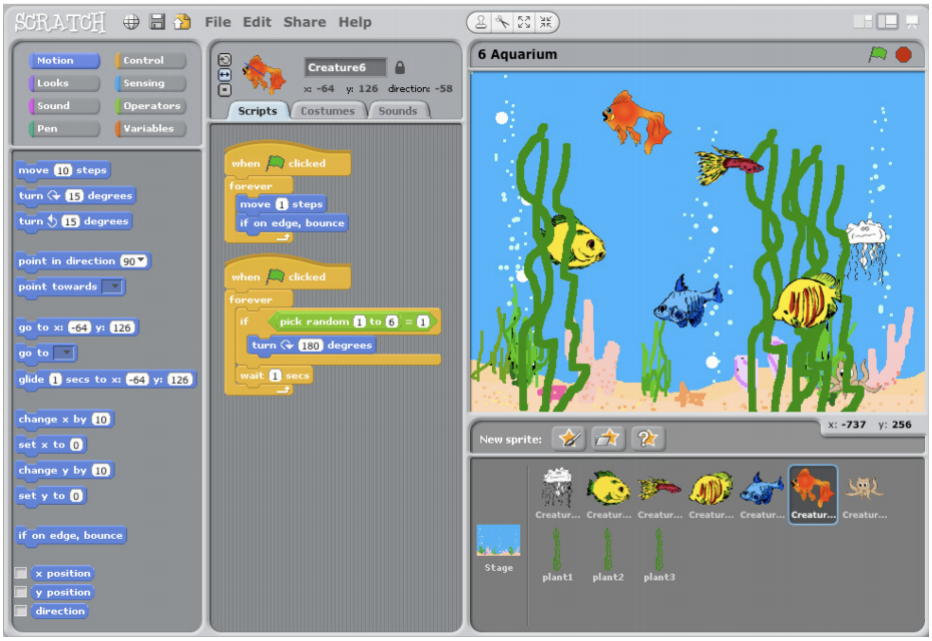

Figura 2.1: Exemplo de programação com *Scratch* [35].

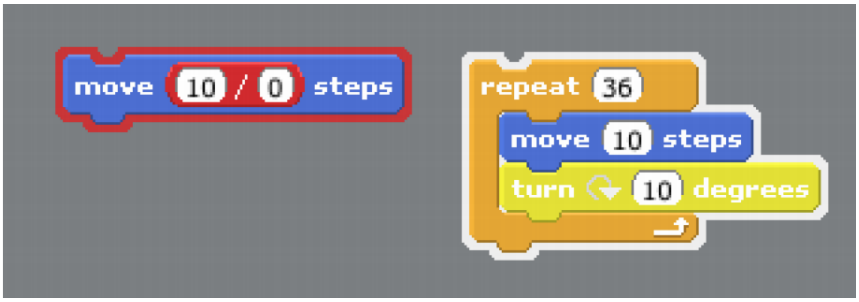

Figura 2.2: Exemplo de erro (esquerda) e de encaixe dos blocos (direita).

Outra abordagem é o *Blockly* [20], uma linguagem inspirada no *Scratch* [35], que possui um editor que se encontra disponível na web.

Assim como o *Scratch* [35], o *Blockly* [20] apresenta do lado esquerdo do editor diversas categorias que dividem os blocos, segundo as suas funções específicas. No centro da interface encontra-se um espaço específico, que possibilita a construção do programa, fazendo *drag and drop* das instruções para este ecrã. No lado direito do ecrã, o programa criado em blocos é traduzido na linguagem em *javaScript* (ver figura 2.3).

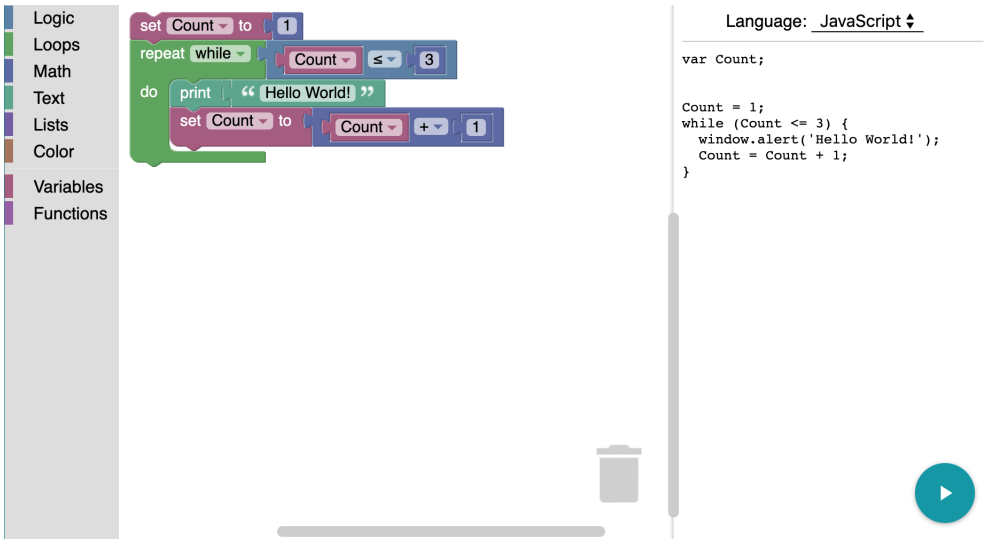

Figura 2.3: Exemplo de programação com *Blockly* [20].

Nesta interface, existem desafios desenhados para a aprendizagem de várias instruções, que permitem ao utilizador chegar a um determinado objetivo. Este tipo de desafios é composto por um mapa, com um caminho amarelo, por onde se desloca um ícone de um pequeno homem cor de laranja, que tem de ser programado, para atingir o objetivo indicado pelo marcador vermelho (ver figura 2.4). Em cada um destes desafios são disponibilizadas um menor número de instruções a serem usadas pelo utilizador, não estando organizadas por categorias, diferindo, desta forma, da interface principal. As funções de cada instrução serão melhor entendidas por quem estiver a aprender, uma vez que o seu uso está limitado.

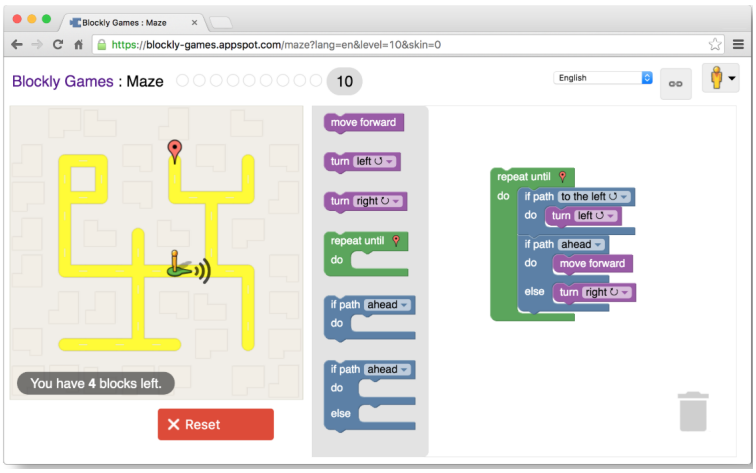

Figura 2.4: Exemplo de desafios com *Blockly* [20].

Inspirada no *Blockly* [20], foi criada uma abordagem mais específica designada de *Ozoblockly* [10]. Esta permite programar através de instruções e posteriormente, descarregar o código produzido para um robô designado *Ozobot* [11], sendo, desta forma,

possível observar o programa efetuado durante a execução do robô.

Em primeiro lugar, é desenvolvido o programa que se deseja que o robô execute, através da colocação das instruções, por movimentos de *drag and drop*, na zona de edição da interface identificada na figura 2.5.

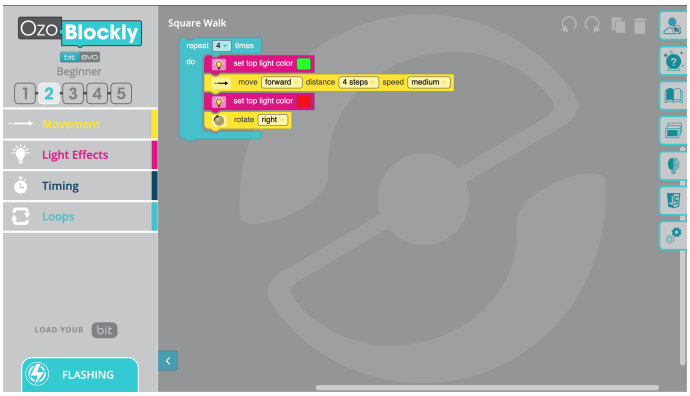

Figura 2.5: Exemplo de programação com *Ozoblockly* [10].

Posteriormente, o robô é posicionado junto à interface circular, apresentada do lado esquerdo da imagem 2.6, de modo a que, através do sensor existente, seja possível calibrar o robô, para que possa descarregar o programa. Em seguida, coloca-se o robô na zona branca com o seu formato e, ao carregar no botão *Load*, é iniciada a transferência do programa, culminando o último passo na execução do robô.

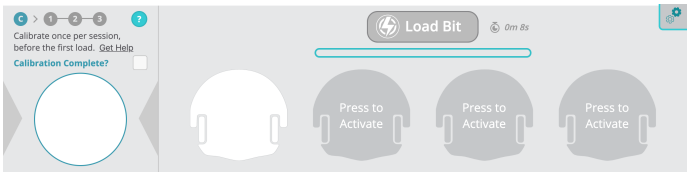

Figura 2.6: Interface que permite descarregar o código para o *Ozobot* [11].

O *Blocks4All* [39] é um sistema desenhado para aprender conceitos de programação destinado tanto a crianças com visão normal, de baixa visão ou invisuais. Este tem a particularidade de ser executado apenas num *IPad* e foi baseado noutros sistemas, nomeadamente o *Scratch* [35] e o *Blockly* [20], em que a criação de programas é feita a partir de blocos representativos de variadas instruções. Na imagem abaixo é possível verificar que existe uma estrutura de áreas de utilização parecida com os sistemas referidos anteriormente. O *Blocks4All* [39] optou por colocar um menu com as instruções no lado esquerdo da interface, a construção do programa ao centro e uma área no lado direito destinada à execução do programa.

Figura 2.7: Esquema de áreas de menus *Blocks4All* [39]

Apesar das semelhanças com outros sistemas, o *Blocks4All* [39] contém algumas diferenças, nomeadamente na interacção com a interface do sistema e pelo facto de permitir a execução de um robô designado *Dash* [3]. Na interação com o sistema há duas abordagens de deslocação dos blocos: a primeira é uma abordagem *audio-guided drag and drop* e a segunda uma *location-first select, select, drop*.

Na primeira abordagem, a instrução é arrastada desde a área onde se encontra e vai sendo dado *feedback* auditivo a cada momento, sobre o local onde está o bloco e se o mesmo deve continuar a ser arrastado, para ser colocado no local correto.

Na segunda abordagem, é escolhida uma localização na zona de construção do programa e depois aparece um menu com as diferentes categorias de blocos possíveis de escolher, sempre acompanhadas com *feedback* auditivo.

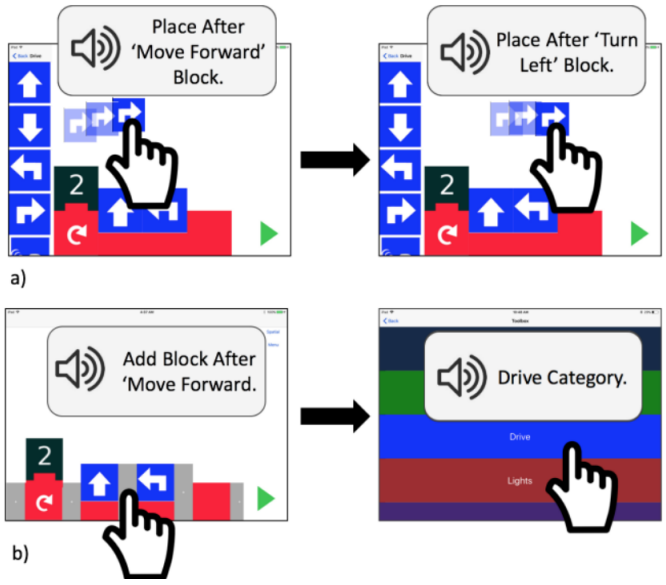

Figura 2.8: a) *audio-guided drag and drop* b) *location-first select, select, drop*.

Na parte da definição da estrutura do código foram também apresentadas duas formas diferentes de execução: a primeira designada de *spatial representation* e a segunda de *audio representation*.

Na primeira abordagem, no que diz respeito aos blocos de repetição (*loop*) e condicionais (*if's*), é colocada em ambos uma peça como bloco de início e outra como bloco de fim. Todas as outras peças que sejam para ser executas dentro destas, são colocadas verticalmente acima, observando-se este tipo de disposição na figura seguinte.

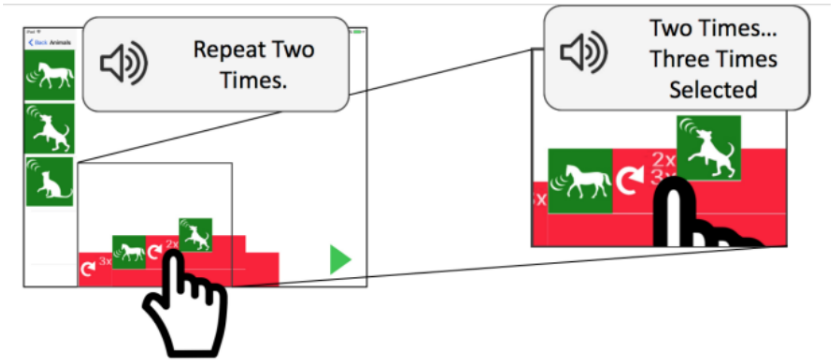

Figura 2.9: *spatial representation*.

Na segunda abordagem, os blocos de repetição e condicionais, permitem que fiquem ligados (*nested*) uns nos outros e, quando selecionados, esta ligação é comunicada verbalmente.

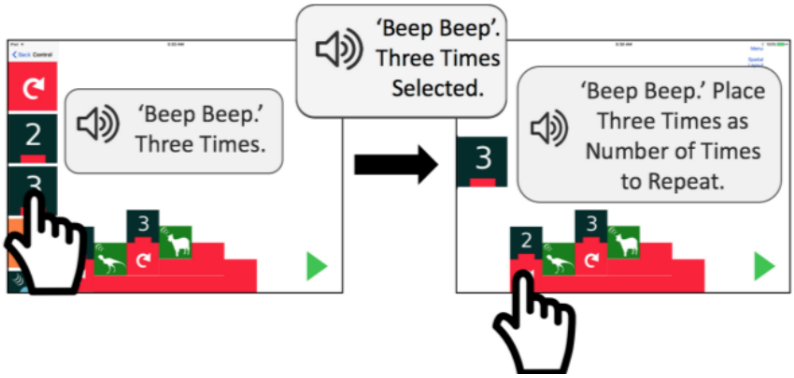

Figura 2.10: *audio representation*

Este sistema permite três tipos de blocos: *operations*, que implica instruções de movimento, *Numbers*, usado em conjunto com blocos de repetição e, por último, *Boolean statements*, usada em conjunto com blocos condicionais.

Um dos sistema que utiliza interfaces gráficas é o *Catroid* [46], que permite programar através de uma linguagem por blocos, do mesmo modo que o *Scratch* [35] num *Android smartphone*, contendo também menus de navegação de instruções e de edição de programas (ver figura 2.11). Este sistema difere do *Scratch* [35], pelo facto de permitir que sejam programados robôs *Lego Mindstorms* [7] e *Drones*, sendo possível a sua movimentação através da passagem de instruções por um protocolo *Bluetooth*.

Figura 2.11: Ambiente de programação do *Catroid* [46].

Uma forma diferente de utilizar um ambiente de programação por blocos é o *codeAttach* [51], em que foi desenvolvida uma aplicação para um dispositivo *Android smartphone* e onde são realizados programas para serem executados por outro dispositivo, desenhado para criar atividades e jogos colaborativos.

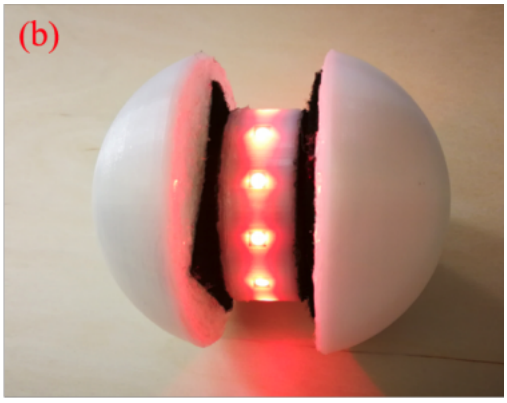

Figura 2.12: Dispositivo com velcro do sistema *codeAttach* [51].

Este dispositivo consiste num cilindro pequeno com duas partes, uma em cima e outra em baixo, cobertas por velcro, no meio destas, existe uma superfície com 9 *RGB LEDs* (ver figura 2.12). Dentro do dispositivo encontram-se um acelerómetro, uma coluna, um vibrador, um modulo *Bluetooth* e uma bateria recarregável. Com a ajuda da aplicação referida anteriormente, é possível programar as luzes, o som, criar efeitos de vibração, introduzindo também conceitos de programação como repetições (*loop*), controlo de fluxo (*if's*) e iniciar a sequência (*start*), como é possível observar na figura 2.13.

Programando este dispositivo, é possível criar atividades interativas, podendo ser jogadas por múltiplos jogadores.

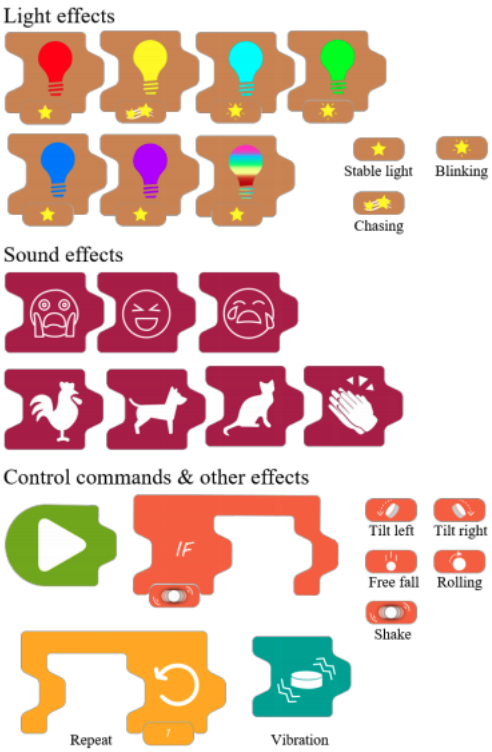

Figura 2.13: Blocos da aplicação do sistema *codeAttach* [51].

2.3 Interfaces Tangíveis

Sistemas com interfaces tangíveis são aqueles em que é necessário uma interação tátil através de blocos ou peças que realizem diferentes funções. O uso destas interfaces é recorrente em sistemas destinados a utilizadores com problemas visuais, nomeadamente cegos e com baixa visão.

A maioria dos sistemas que utilizam interfaces tangíveis contêm um conjunto de blocos ou peças, que apresentam normalmente desenhos tridimensionais alusivos à sua função e que podem conter códigos (TopCodes [27]) que permitem o seu reconhecimento. Apenas alguns dos sistemas têm blocos construídos eletronicamente.

Muitas destas interfaces também são encontradas em sistemas híbridos, que fazem uso de uma interface tangível e de uma interface gráfica.

Uma das abordagens é o *T-Maze* [49] composto por uma interface híbrida em que o sistema se pode dividir em duas partes, uma delas permite a criação de um caminho (labirinto) e na outra é desenvolvido o algoritmo com tangíveis.

Na parte em que se efetua a criação do labirinto, foram colocados inicialmente 20 *TopCodes* [27] para servirem de referência a uma matriz de 10x10, ou seja, existirão 10 códigos no eixo do x e outros 10 no eixo do y, formando, assim, uma espécie de referencial, possibilitando um melhor mapeamento do caminho por parte do sistema.

Para formar o labirinto, as crianças colocam cubos tangíveis que contêm também *TopCodes* [27] e uma imagem referente ao seu significado. Estes blocos determinam diferentes instruções, tais como o início e fim do labirinto, sensores que terão de ser usados posteriormente na parte de desenvolvimento da solução e blocos correspondentes a casas normais, como se pode observar na imagem 2.14.

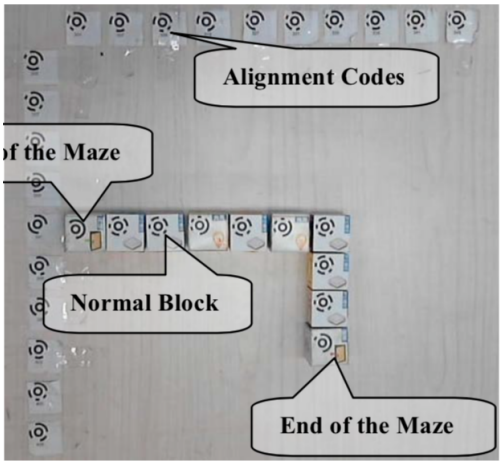

Figura 2.14: Construção de labirinto para *T-Maze* [49].

Em seguida, uma câmara reconhece os *TopCodes* [27] e forma uma representação do labirinto no ecrã do computador, com todos os símbolos associados, como é possível verificar na figura 2.15.

Figura 2.15: Tabuleiro na interface gráfica após o reconhecimento.

Na parte da criação do algoritmo, aproveitou-se a abordagem dos blocos com *TopCodes* [27], para o reconhecimento com a câmara. Estes são representativos das diferentes instruções que permitem indicar o caminho desejado à personagem da interface gráfica. Entre eles existem os blocos de início e fim de sequência, os de direção, os que permitem representar ciclos e os de sensor que são ativados em casas especificas.

Figura 2.16: Blocos de início e fim.

Figura 2.17: Blocos de várias direcções.

Figura 2.18: Blocos para realizar ciclos.

Figura 2.19: Blocos de sensores.

Existe um sistema que não necessita de qualquer tipo de interface gráfica para execução dos tangíveis. Este é designado de *KIBO Robot* [47], que consiste num robô que tem a capacidade de fazer o reconhecimento de códigos de barras que se encontram presentes nos blocos tangíveis. O *KIBO Robot* [47] contém, na parte superior um leitor de código de barras que deteta a sequência realizada com os blocos para que a possa executar, fazendo as diferentes ações descritas em cada bloco.

Figura 2.20: *KIBO Robot* [47].

Figura 2.21: Blocos pertencentes ao conjunto do *KIBO Robot* [47].

Uma outra abordagem é o *Strawbies* [30], um jogo desenvolvido para ser executado num *tablet*. Este jogo processa-se num mundo aberto que é gerado dinamicamente, e no qual se desloca uma personagem chamada *Awbie*, cuja missão é colecionar os morangos que vão aparecendo, tendo que se desviar das árvores que servem de obstáculos à sua movimentação. Se esta tocar em elementos como a água ou morcegos que vão surgindo, o jogo termina.

Para movimentarem *Awbie*, as crianças têm à sua disposição blocos (peças) que são reconhecidas pelo sistema através de *TopCodes* [27], com a ajuda do espelho existente sobre a câmara do *tablet*, designado de *Osmo* [42]. Estes blocos representam ações que permitem ser encaixadas entre si, tais como andar para cima ou para baixo e virar para esquerda ou direita e a execução de repetições através de ciclos. Para determinar o número de vezes em que cada ação é executada, existem peças representativas com vários números.

Para além destas duas categorias, existe também o tornado, a luz e o arco-íris, como funções especiais. O tornado permite apanhar todos os morangos numa determinada área, a luz assusta os ratos que tentam roubar os morangos recolhidos e o arco-íris permite um teletransporte para outro lado do mundo.

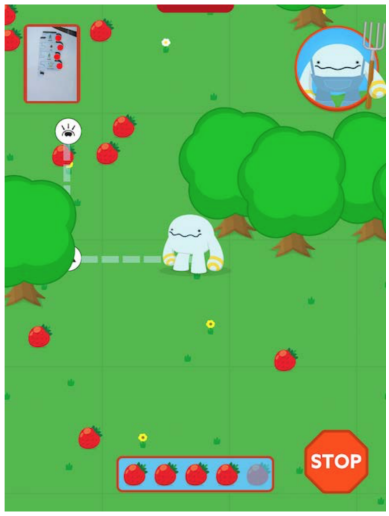

Figura 2.22: *Strawbies* [30] Interface.

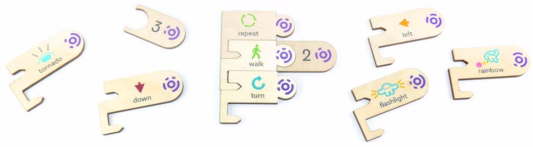

Figura 2.23: Peças com *TopCodes* [27].

Um dos sistemas que usa blocos tangíveis com um objetivo diferente dos sistemas que temos vindo a observar, é o *CETA* [36]. Este é um sistema que promove o ensinamento de conceitos matemáticos como a adição, através de blocos tangíveis que são reconhecidos através de um *tablet*.

Os blocos tangíveis do sistema *CETA* [36] simbolizam os diferentes números através de uma representação específica, pois cada peça contém várias unidades ligadas entre si perfazendo o número a que corresponde, quer dizer que o bloco que representa o número 5 contém cinco unidades, enquanto o bloco representativo do número 2 é mais pequeno, composto apenas por duas unidades. Cada unidade contém um código *Topcode* [27] para que seja detetado pelo dispositivo e cores diferentes para permitir uma melhor distinção. Contém também um íman nas suas pontas para permitir a sua adição e evitando que as crianças os colocassem uns em cima dos outros e, assim, o seu reconhecimento ser afetado. Estas características podem-se observar na figura 2.24.

Figura 2.24: Blocos do sistema *CETA* [36].

A interface gráfica é executada no dispositivo *tablet* referido anteriormente. Esta é responsável pelo jogo, onde a personagem é um robô e que tem como objetivo recolher pregos de um determinado tamanho que a criança tem de decifrar, mas para esse objetivo seja atingido, tem que se ir adicionando blocos até atingir o tamanho desejado pelo jogo. O dispositivo tem na sua câmara um espelho similar ao do *Osmo* [42] que permite colocar os blocos no campo de visão a câmara, permitindo, assim o seu reconhecimento, como é possível observar na figura 2.25.

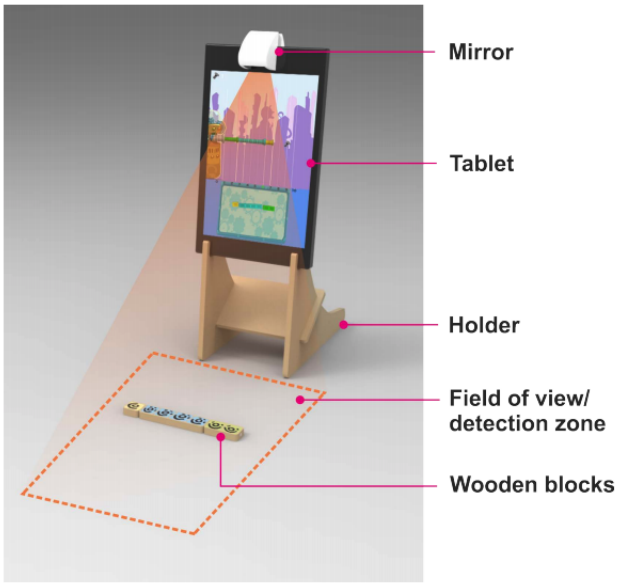

Figura 2.25: conjunto do sistema *CETA* [36].

Existe, ainda, o *iCETA* [43] que corresponde a uma adaptação do *CETA* [36], referido anteriormente. Este sistema utiliza os mesmos blocos do sistema *CETA* [36] que são inspirados pelas *cuisenaire rods* [23] , contendo os blocos pequenas unidades ligadas, que correspondem a um número (de 1 a 5). Para ajudar à organização foi criada uma caixa que permite guardar os blocos (ver figura 2.27).

Figura 2.26: Blocos do sistema *iCETA* [43] organizados na caixa.

iCETA [43] contém ainda um jogo interativo que é executado num computador, e que tem um mágico como personagem que vai realizando feitiços e outras atividades com a ajuda da resolução de desafios que são propostos aos utilizadores.

Figura 2.27: Blocos do sistema *iCETA* [43] organizados na caixa.

Existe um sistema designado de *Tern* [28], que permite através de tangíveis, programar sequências, que posteriormente servem para executar um robô virtual num labirinto virtual. Este labirinto é possível observar na figura 2.28.

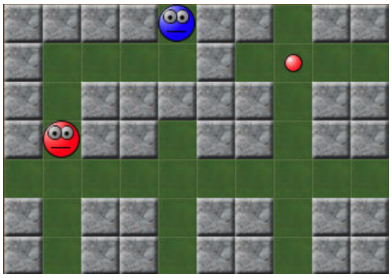

Figura 2.28: Labirinto virtual do *Tern* [28]

Este sistema contém blocos de madeira que encaixam entre si devido ao seu formato, uma vez que se assemelha a uma peça de *puzzle*, em que algumas não são compatíveis com todas as outras peças, devido ao facto de terem formatos diferentes, servindo, desta forma, para dar *feedback* da sua utilização. Cada peça contém um *spotcode* [34] que permite o seu reconhecimento por uma câmara instalada num computador ou *tablet*, e que em seguida permite obter a sequência de modo digital para movimentar robôs digitais no labirinto virtual.

Entre estas peças encontra-se as *start* e stop que marcam o início e o fim da sequência respetivamente, as peças de *move* e *right* que permitem andar em frente e virar para a direita. Por último encontram-se blocos condicionais que permitem executar o programa consoante o estado do robô, ou seja, estes têm um fio que permite determinar o fluxo do programa, dependendo das condições, funcionando como um *goto* ou *jump* em programação. Estas características são possíveis de observar na figura 2.29.

Figura 2.29: blocos tangíveis do sistema *Tern* [28]

Uma outra abordagem é o *Quetzal* [29], uma linguagem que permite a interação com robôs, tendo por objetivo o controlo de *LEGO Mindstorms RCX brick*.

Quetzal [29] contém interface tangível que consiste em peças de plástico que se interligam e representam estruturas, ações e parâmetros. Estas, quando interligadas, formam uma *statement*, uma representação da sequência que permite a programação do robô.

As crianças, com estes blocos ou peças, são capazes de movimentar o robô nas diferentes direções, iniciar o motor, esperar 3 segundos, parar o motor. Depois podem adicionar ou mudar os valores dos parâmetros relacionados com as funções de espera (*wait*) e de alteração de potência do motor, entre outras. Existem também peças fulcrais como o *begin*, *end* e *merge*. No caso do *begin* e *end*, eles servem para marcar o início e fim da *statement*, o *merge* permite a execução de um ciclo.

A ideia fundamental por parte do *Quetzal* [29] é aliar o ambiente de programação por blocos a funções específicas do robô. A grande vantagem deste sistema é que pode ser facilmente adaptado para crianças invisuais, pois com os blocos tangíveis é possível criarem programas através do tato e, para além disso, obter *feedback* auditivo ao longo da execução do robô.

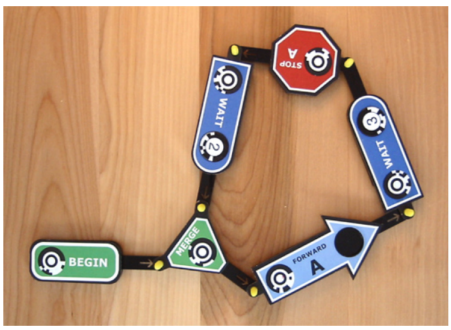

Figura 2.30: Exemplo de *statement* com *Quetzal* [29]

Uma abordagem diferente é o sistema *StoryBlocks* [33]. Este é composto por blocos tangíveis que contêm uma *tag* para ser interpretada por uma biblioteca chamada *reacTIVision* [12] e um desenho saliente, que permite a identificação do bloco pelo tato, correspondendo a uma instrução. Ao contrário da maioria dos sistemas, o objetivo destes blocos é contar uma história, ou seja, cada um deles encaixa num outro, criando uma frase. Os três tipos de blocos que compõem o sistema são: as personagens, as ações e os comandos, aos quais foram dadas formas e encaixes diferentes, com o intuito de serem distinguidos.

Figura 2.31: Blocos Tangíveis do sistema *StoryBlocks* [33].

O objetivo do sistema é ser facultada uma determinada frase e as crianças conseguirem encadear uma sequência de blocos, que após serem reconhecidos pela câmara, seja dado *output* auditivo, com uma frase igual à inicial.

No caso da figura abaixo, a frase que foi recolhida é " O rato comeu o queijo", sendo a personagem o "rato", a ação "comeu"e o comando o "queijo".

Figura 2.32: Exemplo de *statement* do sistema *StoryBlocks* [33].

Outra abordagem é o *Torino* [40] que é uma linguagem de programação, que permite criar um código através da conexão física de *instruction beads* com fios e ajustar parâmetros, usando *onbead mechanisms* para gerar música ou histórias, com a linguagem de *Sonic Pi*.

Este sistema é composto por vários elementos que se interligam e, no centro deste, existe um *central hub*, ao qual vão ser ligados *bead threads* que consistem num conjunto de *beads* ligados entre si, permitindo a metáfora para o termo *thread* usado em computação. O *central hub* contém um *Raspberry Pi* que é responsável por dar energia aos *beads* e identificar a topologia de todas as conexões e, em seguida, poder traduzir para a linguagem *Sonic Pi*. Existe também uma coluna incorporada, para que seja possível ouvir o resultado do programa efetuado.

Cada *bead* é responsável por uma instrução do programa, existindo três tipos de *beads*: o *play*, o *pause* e o *loop* (ciclo). Todos os *beads* têm um formato arredondado, mas o *pause* tem uma forma mais esférica e o *loop* tem os lados mais planos, permitindo, assim, uma melhor diferenciação das funcionalidades de cada *bead*. As cores são diferentes, para que pessoas com baixa visão possam distingui-las melhor.

O que acontece neste sistema é que quando um *bead* se encontra conectado, transmite dados ao seu vizinho. As mensagens são propagadas pela rede de *beads* e, assim que estas atinjam o *Central bead*, ele percebe a topologia da rede, porque as mensagens vão mantendo o conhecimento armazenado no seu percurso. Desta forma, é possível construir um gráfico com a referida topologia e, posteriormente, um *Python scrypt* é executado no *Raspberry Pi*, traduzindo em código de *Sonic Pi* que, por último, permite ouvir os áudios que foram compostos.

Figura 2.33: *Torino* [40] beads (esquerda para a direita): pause (amarelo), *loop* (branco), play (red), hub (quadrado)

Foi desenvolvido um protótipo [44], para um estudo, em que através de blocos tangíveis consegue-se realizar ações com um robô chamado *Dash*. Este é composto por três componentes: blocos tangíveis, um mapa e o robô.

A componente tangível do protótipo utiliza diferentes blocos para a programação do robô, sendo estes representativos de instruções diferentes, que permitem construir o programa que se pretende realizar. Existem, assim, os *Action Blocks* que indicam o movimento, os *Direction Blocks* que apontam a direção do movimento para o robô andar e, por último, o *play block* que permite correr o programa. Para se poder distinguir os diferentes blocos, foram adicionados aos *Action Blocks*, adesivos táteis redondos e botões de diferentes cores e, no caso dos *Direction Blocks*, foi usada uma seta 3D sobre velcro. Uma particularidade dos *Action Blocks* é que estão incorporados com botões que permitem enviar áudio para um dispositivo secundário por meio de *bluetooth* indicando a correspondência da ação.

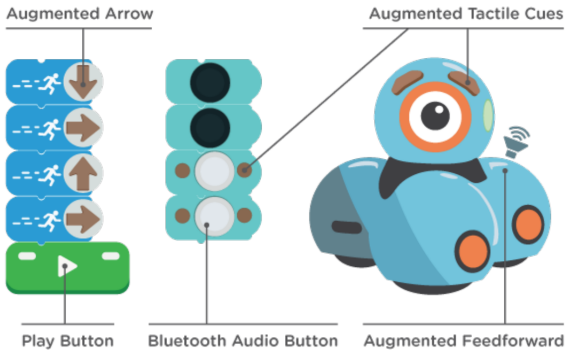

Figura 2.34: blocos tangíveis (esquerda) e robô *Dash* [3] (direita).

O mapa foi desenhado colocando-se numa mesa azulejos quadrados almofadados (33 x 33 cm) com duas cores diferentes e intercaladas entre si, para que as crianças de baixa visão pudessem mais facilmente distingui-los. Cada unidade representa uma casa, estando interligadas entre si, existindo uma união que através do tato possibilita a sua identificação.

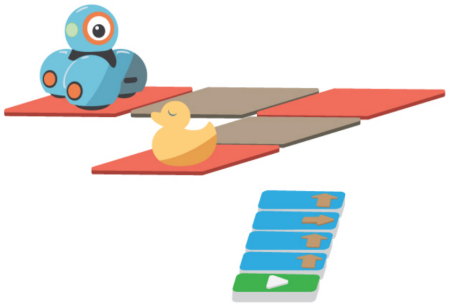

Figura 2.35: Exemplo de mapa do caminho para o robô *Dash* [3] se deslocar

O robô é bastante *playfull* e as crianças acham divertido interagir, tendo sido colocadas umas sobrancelhas com relevo, para que fosse percetível qual a orientação do robô. Nesta

atividade foi efetuada uma abordagem *wizard of Oz*, na qual as crianças colocam as peças e, em simultâneo, um investigador traduz o programa, na plataforma *Blockly* e executa as instruções para o robô se movimentar.

Uma abordagem que tem como objetivo a promoção do pensamento computacional, de forma divertida e *playfull* é o *TACTOPI* [17]. Este sistema consiste num jogo, em que foram desenhados níveis, com o intuito de desenvolver o pensamento computacional, e no qual se controla um robô ("nave") designado de *Micro:bit*, que se movimenta sobre um mapa.

Num estudo [45], foi desenvolvido um sistema que permite a criação de música usando blocos tangíveis e possibilitando a aprendizagem de programação a crianças com deficiências visuais. A ideia consiste na ordenação de blocos, organizando-se numa estrutura algorítmica que produza melodias.

Para criar estas melodias, são necessários diferentes tipos de blocos, distinguindo-se os seguintes:

- bloco *Start*, responsável por iniciar o código.
- bloco *Clear All* que pode ser usado, logo a seguir ao bloco *Start* e que limpa toda a codificação realizada abaixo desse bloco, funcionando como comentário ao código.
- bloco *loop* permite repetições de instruções.
- bloco *Wave Type* que tem duas opções, indicando o tipo de onda que deve ter a melodia, podendo ser *Square* ou *Sine*.
- bloco de *Sound* que executa três instrumentos: violoncelo, piano e harmónio.
- bloco *Beat* que altera a velocidade da música, lenta, moderada ou rápida.
- bloco *Frequency* que possibilita uma alteração da frequência, fraca, média ou alta.
- bloco *Info block* que quando a aplicação se encontra no modo *Info*, informa através de áudio sobre a identificação dos blocos.
- bloco *Run Block* que permite executar a melodia quando terminada.

Estes blocos tangíveis são colocados numa grelha, como é possível observar na figura seguinte e, posteriormente, através de uma aplicação *Android*, desenvolvida para o reconhecimento de cada bloco, é passada, à vez, por cima de cada um destes, identificando os adesivos *NFC*, presentes nos blocos e assim que o *Run Block* é detetado pela aplicação e a melodia começa a ser tocada numas colunas.

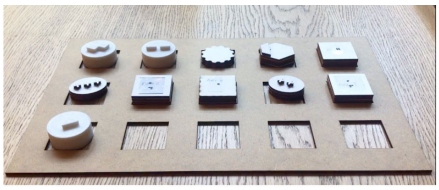

Figura 2.36: Grelha para a colocação de blocos tangíveis.

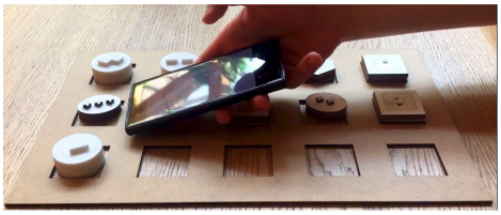

Figura 2.37: Leitura de tangíveis com a aplicação.

Um dos últimos sistemas que explora tangíveis é o *Tip-Toy* [18]. Este sistema foi criado com o objetivo de ser um *open-source toolkit* de baixo custo e que permitisse desenvolver o pensamento computacional de crianças com deficiências visuais.

O *Tip-Toy* [18] é composto por um acrílico em forma de "U"que permite a colocação de uma câmara num determinado ângulo, para que seja possível reconhecer *QR codes* que se encontram presentes em blocos tangíveis (ver figura 2.38).

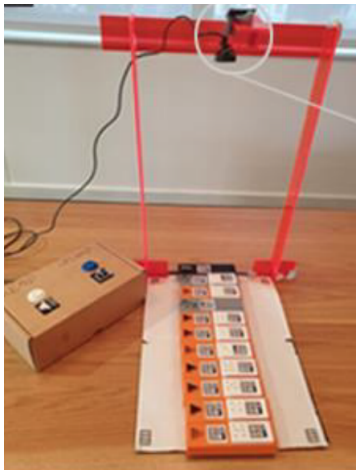

Figura 2.38: Visão geral do sistema *Tip-Toy* [18].

A colocação de blocos tangíveis na área branca que é possível observar na figura 2.38, tem como objetivo a reprodução de diferentes sons. Entre os blocos que compõem o sistema, encontram-se blocos de repetição (*Loops*), um bloco de *Play* e um bloco de variável (*Variable Block*).

Existe ainda uma particularidade do sistema, o facto de ter uma caixa com dois botões, um de cor azul para poder fazer *upload* da sequência para o sistema de modo a ser executada e um botão branco que permite que seja lida a sequência, funcionando como um

debugger para a criança (ver figura 2.39).

Figura 2.39: butões de *upload* e *read* do sistema *Tip-Toy* [18]

2.4 Interfaces de Voz

Interfaces de voz possibilitam interação verbal com o sistema, para que este realize as funções desejadas.

Voxtopus [38] é um sistema que apoia atividades pedagógicas inclusivas, estruturado em torno de um dispositivo chamado *Amazon Echo*, que fornece um assistente integrado designado de *Alexa* e possui também alguns controladores físicos ligados. Este foi desenhado com o objetivo de carregar para o dispositivo, conteúdo de matérias lecionadas na escola, possibilitando a realização de sessões de pergunta e resposta. Nestas, o controlador de voz analisava os conteúdos carregados para formular questões, que eram dirigidas aos alunos e as sessões eram colaborativas e sempre com a presença de alunos com deficiências visuais.

Os controladores físicos foram analisados com o objetivo de permitirem perguntas de escolha múltipla, podendo através dos botões que têm integrados, escolher a opção que fosse mais desejada.

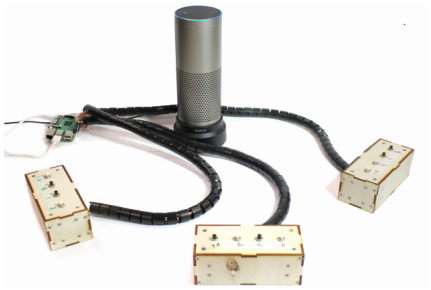

Figura 2.40: Voxtopus [38].

Uma abordagem diferente e que utiliza interfaces de voz é o *TurtleTalk* [31]. Este é um sistema *web* que opera em qualquer dispositivo, composto por um ecrã e colunas e pretende desenvolver o pensamento computacional, no qual são dadas instruções verbais.

O sistema apresenta uma componente gráfica, onde aparece uma personagem principal que é uma tartaruga, um caminho por onde esta se irá deslocar e um objeto indicando o

alvo final a atingir.

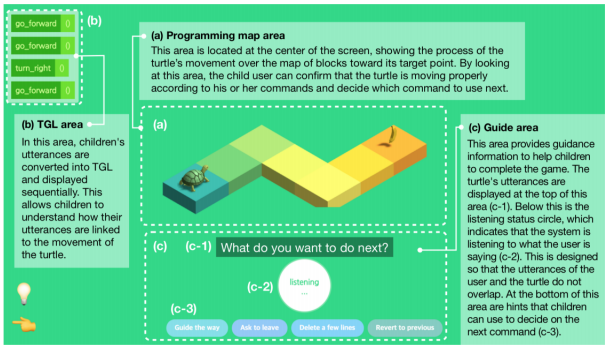

Figura 2.41: Interface gráfica do *TurtleTalk* [31].

A tartaruga vai movimentar-se após serem recolhidas as instruções para o sistema, sendo estas dadas verbalmente, por meio de uma estratégia conversacional, ou seja, através de um conjunto de questões que são colocadas ao utilizador, e às quais este terá de responder. Assim, o sistema percebe quais as direções que a personagem tem de executar.

Em termos de aprendizagem, são incutidos os conceitos de sequência e iteração. O primeiro baseia-se em instruções simples que são adquiridas uma de cada vez e o segundo obtém-se através do uso de ciclos que permitem efetuar a repetição de instruções.

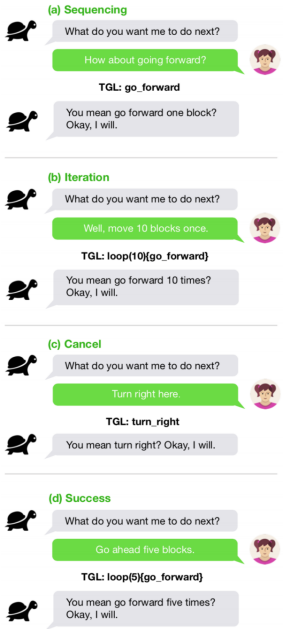

Figura 2.42: Estratégia conversacional do *TurtleTalk* [31].

2.5 Discussão

Nesta secção, são apresentadas as características dos vários sistemas mencionados nas secções anteriores. De uma maneira generalizada, é possível verificar os tipos de sistemas existentes e quais as características que podemos inovar.

Sistemas & Características		Características												
		Interfaces			Dispositivos			Conceitos Computacionais						
		Tangível	Gráfica	Reconhecimento Voz	Tablet	Computador	Smartphone	Sequência	Ciclos	Condicionais	Eventos Paralelos	Operadores	Dados (valores)	Outras
Sistemas	Scratch [35]	Não	Sim	Não	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
	Blockly [20]	Não	Sim	Não	Não	Sim	Não	Sim	Sim	Sim	Sim	Sim	Sim	Sim
	Ozoblockly [10]	Não	Sim	Não	Não	Sim	Não	Sim	Sim	Sim	Sim	Sim	Sim	Sim
	Blocks4All [39]	Não	Sim	Não	Sim	Não	Não	Sim	Sim	Sim	Não	Sim	Não	Sim
	T-Maze [49]	Sim	Sim	Não	Não	Sim	Não	Sim	Sim	Não	Não	Sim	Não	Sim
	Strawbies [30]	Sim	Sim	Não	Sim	Não	Não	Sim	Sim	Não	Não	Sim	Não	Sim
	Quetzal [29]	Sim	Não	Não	Não	Não	Não	Sim	Sim	Sim	Sim	Sim	Não	Sim
	StoryBlocks [33]	Sim	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não	Sim
	Torino [40]	Sim	Não	Não	Não	Não	Não	Não	Sim	Não	Sim	Sim	Não	Sim
	Protótipo com Dash [44]	Sim	Não	Não	Não	Não	Não	Sim	Sim	Sim	Não	Sim	Não	Sim
	Protótipo Música [45]	Sim	Sim	Não	Não	Não	Não	Não	Sim	Não	Não	Não	Não	Sim
	Voxtopus [38]	Não	Não	Sim	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não
	TurtleTalk [31]	Não	Sim	Sim	Não	Sim	Não	Sim	Sim	Não	Não	Sim	Não	Não
	Tip-Toy [18]	Sim	Não	Não	Não	Não	Não	Não	Sim	Não	Não	Não	Não	Sim
	Tern [28]	Sim	Sim	Não	Não	Sim	Não	Sim	Sim	Sim	Não	Não	Não	Não
	CETA [36]	Sim	Sim	Não	Sim	Não	Não	Não	Não	Não	Não	Não	Não	Sim
	iCETA [43]	Sim	Sim	Não	Não	Sim	Não	Não	Não	Não	Não	Não	Não	Sim
	CodeAttach [51]	Não	Sim	Não	Sim	Não	Não	Não	Sim	Sim	Não	Não	Não	Sim
	Catroid [46]	Não	Sim	Não	Não	Não	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
	Kibo [47]	Sim	Sim	Não	Não	Sim	Não	Sim	Não	Não	Não	Não	Não	Sim

Tabela 2.1: Características de sistemas (parte 1)

Sistemas & Características		Características									
		Práticas Computacionais				Mapas		Uso Robôs	Testagem (Nº da amostra)		
		Experimentação & iteração	Testes & Debug	Reutilização	Abstração & Modularização	Virtuais	Físicos		Cegos	Baixa Visão	Visão Normal
Sistemas	Scratch [35]	Sim	Sim	Sim	Sim	Não	Não	Não	N/A	N/A	N/A
	Blockly [20]	Sim	Sim	Sim	Sim	Sim	Não	Não	N/A	N/A	N/A
	Ozoblockly [10]	Sim	Sim	Sim	Sim	Não	Não	Sim	N/A	N/A	N/A
	Blocks4All [39]	Sim	Não	Não	Não	Não	Não	Sim	1	4	0
	T-Maze [49]	Sim	Sim	Não	Não	Sim	Sim	Não	N/A	N/A	N/A
	Strawbies [30]	Sim	Não	Não	Não	Sim	Não	Não	N/A	N/A	N/A
	Quetzal [29]	Sim	Não	Não	Não	Não	Não	Sim	N/A	N/A	N/A
	StoryBlocks [33]	Sim	Não	Não	Não	Não	Não	Não	5		11
	Torino [40]	Sim	Sim	Não	Não	Não	Não	Não	5	1	4
	Protótipo com Dash [44]	Sim	Sim	Não	Não	Não	Sim	Sim	2	6	0
	Protótipo Música [45]	Sim	Sim	Não	Não	Não	Não	Não	14		0
	Voxtopus [38]	Sim	Não	Não	Não	Não	Não	Não	N/A	N/A	N/A
	TurtleTalk[31]	Sim	Sim	Não	Não	Sim	Não	Não	N/A	N/A	N/A
	Tip-Toy [18]	Sim	Sim	Não	Não	Não	Não	Não	1	3	6
	Tern [28]	Sim	Sim	Não	Não	Sim	Não	Não	N/A	N/A	N/A
	CETA [36]	Sim	Não	Não	Não	Não	Não	Não	19		
	iCETA [43]	Sim	Não	Não	Não	Não	Não	Não	N/A	N/A	N/A
	CodeAttach [51]	Sim	Sim	Não	Não	Não	Não	Não	7		
	Catroid [46]	Sim	Sim	Sim	Sim	Não	Não	Sim	N/A	N/A	N/A
	Kibo [47]	Sim	Sim	Não	Não	Não	Não	Sim	32		

Tabela 2.2: Características de sistemas (parte 2)

Nas tabelas 2.1 e 2.2 observamos as características dos diferentes sistemas e nelas estão assinaladas as que são necessárias focar. A cor verde corresponde às características que cada um dos sistemas possui e a cor vermelha às não existentes.

Nas colunas abrangidas pelas interfaces é possível observar que existem mais sistemas que utilizam interfaces tangíveis e interfaces gráficas, no entanto desses sistemas apenas dois permitem o reconhecimento de voz (*Voxtopus* [38] e *TurtleTalk* [31]).

Nas colunas sobre os dispositivos, é possível observar que o número de sistemas com interfaces gráficas é igual ao número de sistemas assinalados a verde, mas encontram-se

bastante dispersos entre os 3 tipos de dispositivos. É também possível observar que dos sistemas que contêm interfaces gráficas, são maioritariamente executadas em computador.

Nas colunas dos conceitos computacionais são apresentados vários tipos de instruções. A sequência refere-se a instruções básicas de movimentação (frente, esquerda, direita, etc.), que é muito comum a sua utilização, tal como é possível observar na tabela. Os ciclos permitem executar instruções de sequência de forma repetida, sendo utilizados por quase todos os sistemas. As instruções condicionais encontram-se relacionadas com a decisão de ações baseadas em condições, normalmente *boolenas*, no entanto nem todos os sistemas as adotam pelo grau de complexidade dos programas para determinadas idades. Eventos paralelos implica que se encontrem a executar dois programas ao mesmo tempo (*threading*), não sendo adotado pela maioria dos sistemas. Os Dados encontram-se relacionados com o armazenamento e a alteração de valores durante a execução de programa, este conceito é pouco utilizado pelos sistemas. Os operadores estão relacionados com noções matemáticas, como a utilização de números, que é uma prática comum no uso de ciclos para determinar o número de repetições. Existem ainda outras instruções que são menos relevantes para a aprendizagem de conceitos, mas podem ser interativas para as crianças, tal como por exemplo, um bloco despoletar um áudio ou uma dança.

Nas colunas das práticas computacionais, encontram-se algumas funcionalidades que os sistemas permitem durante a sua utilização. A experimentação e iteração referem-se à possibilidade de executar programas através de tentativa e erro, podendo melhorar iterativamente e todos os sistemas permitem fazê-lo. Testes e *debug* está relacionado com a obtenção de *feedback*, após o programa ser testado ou executado, a maioria dos sistemas suporta esta prática. Reutilização relaciona-se com a possibilidade de os sistemas guardarem os programas já realizados e poderem voltar a ser usados, sendo esta uma funcionalidade pouco comum. A abstração e modularização está ligada à criação de funções para permitir lidar com complexidade dos programas que, apesar de ser muito o ponto de interesse na aprendizagem do pensamento computacional, não se encontra muito presente nos sistemas apresentados na tabela.

Nas colunas relativas aos mapas é possível observar que quer os virtuais, ou os físicos têm um número reduzido.

Na coluna do uso de robôs, tal como a dos mapas, têm poucos sistemas que os utilizam, no entanto é preciso salientar que existe apenas um sistema (protótipo com o *Dash* [44]) que contém estas duas características (uso de robô e mapa físico).

Na coluna da testagem é de notar que existe informação que não se encontra disponível (N/A), não querendo dizer que não houve grupos de teste, mas que em alguns artigos não foram indicados a que grupos pertenciam as pessoas. Na testagem existem colunas que se encontram ligadas,isto é, existiam no seu conjunto pessoas daqueles grupos, não tendo sido feita qualquer tipo de divisão. Apesar de não serem sempre referidos os grupos a que pertencem a sua amostra de teste, existem alguns sistemas que foram testados com

crianças cegas e de baixa visão.

Robôs & Características		Características	
		Dimensões (largura)	Custo (euros)
Robôs	Ozobot Bit [11]	2,54 cm	30,00
	Dash [3]	19,51 cm	263,03
	DOC (Clementoni) [4]	27,80 cm	33,90
	Makeblock (mbot) [9]	9,00 cm	99,99

Tabela 2.3: Características de alguns robôs

Foi realizada uma recolha de informações de alguns dos robôs presentes no mercado, sendo analisados apenas dois parâmetros, a dimensão (largura) e o custo. Ao observar a tabela 2.3, verifica-se que se encontra destacado o robô *Ozobot Bit* [11], uma vez que é mais económico, facilitando assim a sua aquisição, e possui menores dimensões, permitindo a sua movimentação em espaços reduzidos.

Após a análise efetuada consideramos existirem algumas características que são desejadas num sistema que permita, a que crianças cegas aprendam conceitos de programação:

1. O sistema deve ter uma interface que possibilite o reconhecimento de voz, para além das interfaces tangível e gráfica.
2. O sistema deve ser executado num *tablet* ou *smartphone*.
3. O sistema deve permitir a construção de um mapa físico.
4. O sistema deve fazer uso de um robô, para que seja possível a sua deslocação sobre o mapa físico anteriormente referido.

Capítulo 3

Desenho do Sistema

3.1 Abordagem Proposta

Como abordado no capítulo anterior, o desenvolvimento do pensamento computacional encontra-se diretamente relacionado com a prática de construção de programas, através da interligação de conceitos de programação [50].

Embora existam alguns sistemas que se focam no ensinamento de práticas e conceitos de programação [35, 20, 10], ainda permanecem algumas limitações na obtenção deste tipo de informação, que não se encontra acessível para pessoas com limitações visuais, como pessoas cegas ou de baixa visão. Estes conhecimentos, podem começar a ser incutidos em crianças, de modo a assimilarem os conceitos de uma forma mais consistente e facilitar o processo de aprendizagem ao longo do tempo.

Atualmente, já existem sistemas que propõem atenuar as dificuldades visuais desde idades muito jovens, fornecendo mecanismos, através dos quais é facilitada a aquisição de conceitos de programação. [48, 30, 44].

Os sistemas promovem o ensinamento de conceitos de programação, através de diferentes interfaces, sendo usadas maioritariamente as gráficas e tangíveis, permitindo divulgar, por um mecanismo de blocos, as várias instruções usadas na construção de diferentes programas. Algumas destas abordagens permitem o encadeamento de instruções complexas que conseguem envolver a programação de robôs virtuais ou físicos, fazendo com que o sistema se torne mais apelativo e dinâmico para as crianças.

Nesta sequência, definiu-se a construção de um sistema que satisfaz alguns requisitos que pensámos serem essenciais:

- O sistema deve conter uma interface gráfica que possibilita ao utilizador interagir, sendo executada num *Smartphone* ou *tablet*.

- Para que crianças cegas ou com baixa visão consigam aprender as instruções essenciais à construção de um programa é necessário, que seja desenvolvida uma solução que incorpore uma interface tangível.

- De modo a comparar a aprendizagem de instruções por métodos diferentes, o sistema deve ter uma interface de voz, que permita o reconhecimento da fala.

- Para que o sistema se torne mais interativo, dinâmico e permitir um *feedback* tátil, deve existir um robô que seja programável pelas crianças.

- O sistema deve proporcionar uma plataforma para a construção de qualquer caminho, criando um mapa físico tridimensional e tangível, com o objetivo de o robô se deslocar no meio dos seus obstáculos.

- O sistema deve permitir *feedback* auditivo relativo aos movimentos que o robô está a executar.

3.2 Cenários de Utilização

Nesta secção, encontram-se alguns cenários que permitem demonstrar, descrever e esclarecer o funcionamento do sistema, nos seus diferentes modos de interação (modo por blocos e modo por voz) e nos modos de jogo (modo *feestyle* e modo *accelarated*) possíveis na aplicação.

Os vários cenários integram dois intervenientes, uma criança com deficiências visuais, podendo ser cega ou ter baixa visão e um educador.

3.2.1 Cenário 1 - Utilização de Blocos Tangíveis

O António tem 5 anos e ficou cego devido a uma doença progressiva que foi afetando a sua visão ao longo do tempo.

o João, pai do António, considera que o filho deve desenvolver as suas capacidades cognitivas de maneira a que este, não se sinta estigmatizado no futuro, considerando utilizar um sistema que permita ultrapassar barreiras.

O João começa por ler o guião que explica a montagem do sistema. Segue atentamente as indicações, colocando o tablet no tripé e, em seguida, centra-o relativamente ao tabuleiro verde, de modo a ficar com a câmara paralela a este. Em seguida, verifica que é necessário a construção de um caminho com Legos, específico para o 1º nível do jogo. Entretanto, o João percebe que tem de montar um tabuleiro, com 4 casas para a frente e inicia a montagem deste ao analisar as cores de cada Lego, encaixando neste mesmo tabuleiro a casa branca, que indica o início do caminho, quatro peças vermelhas de cada lado para os obstáculos e uma casa final da cor azul. Por último, encaixa as casas verdes lisas entre os obstáculos, de modo a ser possível a deslocação do robô.

Após a montagem, o Pai e o António conversam sobre a disposição do tablet, do tabuleiro e do caminho que têm à sua frente.

Posteriormente, o João posiciona uma caixa com um conjunto de blocos tangíveis que vão permitir a programação do robô e o pai de António explica ao filho que cada peça corresponde a uma instrução diferente e que de forma a perceber qual a correspondência, deve verificar através do tato qual o relevo que está embutido no bloco.

Depois de António entender a disposição de todos os elementos que compõem o sistema, o pai abre a aplicação para iniciar o jogo, coloca as iniciais do filho para a criação de um utilizador, carrega no botão correspondente ao primeiro nível e, neste momento, tem a câmara ligada, para o início do jogo.

A aplicação vai dando indicações de como proceder, para que seja possível jogar. Antes de construir o programa, António perceciona o caminho através das diferentes texturas dos Legos e, assim que pensa ter descoberto a solução, começa por colocar os blocos tangíveis na placa que tem do seu lado direito. Entretanto, tenta perceber qual o bloco que permite andar para a frente, pega num destes blocos, mas como não tem a certeza se é o bloco correto, então pressiona o tablet durante algum tempo, até que a aplicação lhe dê a seguinte indicação: "A peça corresponde a ir para a frente".

Agora que o António já escolheu as peças corretas, coloca-as verticalmente todas seguidas, finalizando com a peça de "play". Carrega no tablet uma vez, tal como tinha sido instruído pela aplicação. Esta reconhece a sequência realizada pelo António e pede-lhe que efetue os passos para executar o robô. Coloca-o na casa de partida e inicia-o ao mesmo tempo que a aplicação vai dizendo, quais as direções a tomar: "vamos para a frente"é dito 5 vezes, seguido de "chegou ao objetivo", finalizando assim o 1º nível.

3.2.2 Cenário 2 - Reconhecimento de Voz

A Vanessa é mãe da Maria, uma criança de 6 anos que nasceu com diabetes melitos tipo 1, causando retinopatia grave, por isso apresenta muito baixa visão.

Vanessa tem formação em engenharia informática e, como tal, sabe as potencialidades que a aprendizagem de programação pode ter a nível cognitivo, desejando que a filha tenha acesso a conhecimentos relacionados com a sua área de formação, tendo adquirido o sistema, para que a filha pudesse aprender tais conceitos.

Vanessa começa por montar o sistema como ilustrado no guião e, após a montagem do tablet no tripé, encaixa as peças Lego para formar o caminho no tabuleiro verde e realiza todos os passos da aplicação para iniciar o nível 1. Entretanto a aplicação indica verbalmente como deve iniciar o nível e Maria começa por percecionar o caminho através das peças de Lego que têm diferentes texturas, percebendo que este tem disponível 4 casas até ao final.

Maria sente que se encontra pronta para iniciar o nível, então carrega uma vez no ecrã, onde lhe são efetuadas um conjunto de questões de modo a obter a sequência que quer realizar. A aplicação começa por perguntar:

Aplicação: - Achas que deva ir para a frente ou virar à direita ou virar à esquerda?

Maria: - Frente.

Aplicação: - Mais alguma instrução?

Maria: - Sim.

(Este diálogo é repetido 4 vezes até chegar ao caso descrito em baixo)

Aplicação: - Mais alguma instrução?

Maria: - Não.

Aplicação: - Então diz terminar.

Maria: - Terminar.

Em seguida, a aplicação indica verbalmente para executar o robô e Maria coloca na casa de partida o robô e inicia-o. Enquanto este se desloca sobre o caminho, vai sendo enunciado pela aplicação as direções que está a tomar a cada momento. "Vamos para a frente"é indicado 5 vezes e por último "chegou ao objetivo", indicando que o nível foi terminado com sucesso.

3.2.3 Cenário 3 - Modo *Freestyle*

O João é uma criança de 5 anos que é cega desde nascença. A Ana é sua mãe e já adquiriu o sistema há algum tempo e como o seu filho já realizou com sucesso todos os desafios, pensa que deve dificultar um pouco.

Para isso, Ana começa por montar o sistema como o habitual, coloca o tablet no tripé, centra o tabuleiro verde para o encaixe de Legos e imagina o que seria um caminho complexo, para o seu filho realizar.

Após Ana ter decidido e montado o caminho, o João começa por percecionar o tabuleiro com o objetivo de entender o caminho possível.

A sua mãe abre a aplicação, insere os dados de utilizador e escolhe o modo de freestyle, que permite detetar qualquer tabuleiro. Em seguida a aplicação indica, que para o modo por voz carrega-se uma vez no ecrã e para o modo por blocos ou tangíveis, carrega-se duas vezes no ecrã e, a partir desse momento, o João pode escolher um dos modos e realizar um novo nível, com o caminho que foi construído pela mãe, executando tudo o resto normalmente.

3.2.4 Cenário 4 - Modo *Accelarated*

O António é uma criança de 10 anos que ficou cega devido a uma doença degenerativa que afetou a sua visão. A Mariana é sua mãe, e tendo na sua posse o sistema há algum tempo, o seu filho António teve a oportunidade de experimentar quase todos os modos do jogo, sentindo que pode desenvolver ainda mais as suas capacidades.

Para melhorar estas capacidades, a sua mãe incentiva-o a experimentar o modo accelarated, que é mais complexo ao nível da interação, quando são dadas as instruções.

Mariana monta o sistema todo, dispondo o mapa com 3 casas para frente e 3 para a direita. Carrega no botão correspondente ao modo acelerated e, neste momento, o ecrã do tablet mostra a câmara ligada, permitindo que se inicie o jogo.

O António inicia o jogo carregando uma vez no ecrã. Posteriormente a aplicação inicia a seguinte interação:

Aplicação: Indica a sequência que queres executar?

António: Vamos para Frente, depois para a frente outra vez e continua em frente, vira à direita, vai em frente, depois continua em frente, e mais uma vez para frente.

Aplicação: Recebido a sequência frente, frente, frente, direita, frente, frente, frente, confirmas a sequência?

António: Sim!

Aplicação: Sequência confirmada!

Em seguida o António calibra o robô e este recebe a sequência. O robô é executado, começando a movimentar-se ao mesmo tempo que a aplicação dá o feedback auditivo sobre as instruções que este está a realizar, indicando que o António chegou ao objetivo.

3.3 Prototipagem

Nesta secção, são introduzidos e apresentados protótipos com diferentes graus de detalhe, que foram efetuados segundo os requisitos anteriormente referidos, servindo como guia para o desenvolvimento do sistema. Estes devem ser observados como parte de um processo iterativo que possibilitou chegar a uma solução final do protótipo.

3.3.1 Protótipo de Baixa Fidelidade

Nesta subsecção, apresenta-se o protótipo de baixa fidelidade, realizado no contexto deste projeto. Este tipo de protótipo é efetuado com o intuito de obter uma solução do funcionamento geral do sistema, com base nos requisitos definidos, não sendo muito pormenorizado e que, normalmente, é considerado como um esboço desenhado à mão.

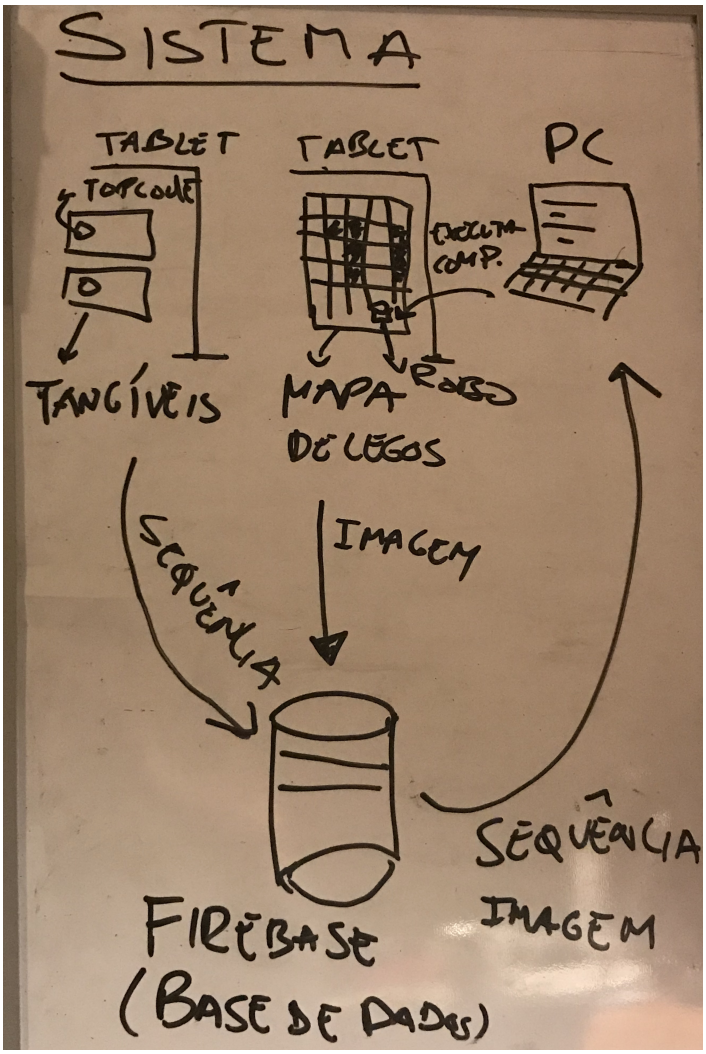

Figura 3.1: Protótipo de baixa fidelidade.

A figura 3.1 mostra o primeiro protótipo do sistema elaborado num quadro branco, com algumas componentes muito diferentes do protótipo final. Nesta figura é apresentada uma visão generalizada do sistema, que corresponde a um mapa físico que permite o encaixe de *Legos* formando um caminho e, que através da programação com blocos tangíveis possibilita a criação de sequências, que fazem movimentar um robô sobre o caminho construído.

Na figura 3.1 é possível não só analisar o sistema de uma forma geral, mas também identificar as várias componentes que o estruturam, contribuindo para a visão referida anteriormente. Estas dividem-se em quatro componentes diferentes: uma componente de blocos tangíveis, uma componente da construção do mapa físico, outra que permite a receção da sequência, sendo necessário um computador e uma última que corresponde à componente de base de dados.

A componente de blocos tangíveis é possível observá-la a partir do lado esquerdo, na parte de cima da imagem, onde os blocos se encontram representados por retângulos com

a legenda assinalada na figura com a palavra "Tangíveis".

A componente de construção do mapa físico apresenta-se na imagem, na parte superior, ao centro, representada por uma grelha que tem alguns quadrados preenchidos a cor preta com a legenda "Mapa de Legos", ilustrando os *Legos* representativos dos obstáculos que permitem definir o caminho livre entre estes. Consegue-se ainda observar que o robô está posicionado em cima desta grelha, com a forma de um cilindro muito pequeno, contendo por baixo a legenda "Robô".

A componente de base de dados encontra-se na parte de baixo da imagem, correspondendo ao cilindro com a legenda "Base de dados". Esta componente era responsável pelo armazenamento das sequências produzidas com os tangíveis, fazendo a ligação entre duas componentes: a dos blocos tangíveis reconhecidos pelo *tablet* e a do computador.

A componente do computador está presente na parte superior do lado direito com a legenda "PC"(*Personal Computer*) e cujo o objetivo era receber a sequência dos tangíveis e possibilitar a sua transferência para o robô, de modo a que este pudesse ser executado através da linguagem *Python*, uma vez que não tínhamos ainda contemplado a possibilidade de executar esta linguagem, noutros dispositivos como o *tablet* ou *smartphone*.

3.3.2 Protótipo de Média Fidelidade

Nesta subsecção, é apresentado o protótipo de média fidelidade. Este é um tipo de protótipo intermédio, encontrando-se numa fase posterior ao de baixa fidelidade e anterior ao protótipo final, tendo sido criado com o objetivo de mostrar mais detalhadamente o que não foi observado no protótipo anterior e com uma visão mais precisa do sistema final. Por ser mais detalhado normalmente é desenhado recorrendo a ferramentas de software mais tecnológicas.

A figura 3.2 representa o protótipo de média fidelidade elaborado, sendo possível verificar que existem grandes diferenças relativamente ao protótipo de baixa fidelidade, principalmente em relação ao detalhe de algumas componentes.

Na imagem podemos observar que a componente do mapa físico foi enriquecida com cores diferentes, com a ideia de que a aplicação distinguisse os vários *Legos*, necessários à montagem do caminho. A cor do tabuleiro é verde, por ser a cor mais comum no mercado, a cor branca foi definida para a casa inicial, a cor vermelha para os obstáculos e a cor azul para o final.

A componente dos blocos tangíveis encontra-se mais detalhada, pois foi colocada uma placa, que na realidade é a tampa que cobre a parte superior da caixa das peças e que possibilita o encaixe destas, numa sequência vertical, por conter uma superfície própria para a sua colocação.

Estes blocos contêm um desenho geométrico que corresponde a uma superfície com relevo e uma forma que permite a crianças cegas identificar a instrução de cada bloco. Possuem ainda um circulo representando códigos de uma biblioteca visual, com o objetivo

de serem reconhecidos pela aplicação e a caixa de blocos presente na imagem, para os armazenar.

O *tablet* contém uma aplicação chamada *OzoUniverse* que permite a interação da criança com o sistema.

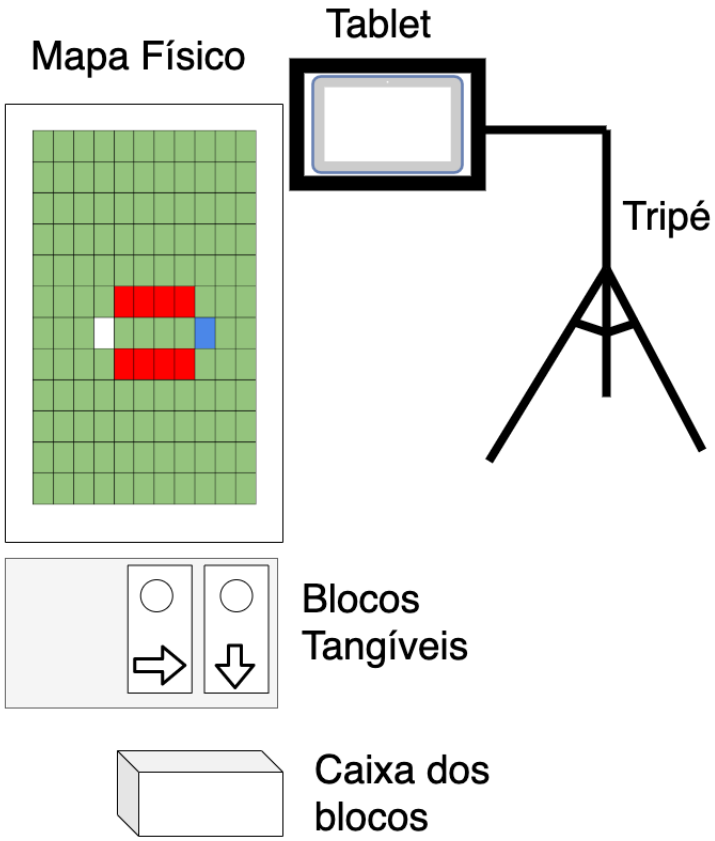

Figura 3.2: Protótipo de média fidelidade.

3.3.3 Protótipo de Alta Fidelidade

Nesta subsecção, é apresentado o protótipo de alta fidelidade que inclui a tecnologia desenvolvida, com base nos protótipos anteriores, exibindo um tipo de interação física, representando uma versão muito próxima do protótipo final.

Nas figuras 3.3 e 3.4 é possível ter uma visão geral do protótipo de alta fidelidade, sendo possível identificar componentes que estavam nos protótipos das secções anteriores. Assim, consegue-se observar algumas componentes, nomeadamente o *board* que permite o encaixe de *legos*, as diferentes cores de cada *Lego*, a placa branca que serve de plataforma para a colocação dos tangíveis, a caixa para guardar as peças e o *tablet* suportado pelo tripé, que contêm a aplicação que permite a interação com o sistema.

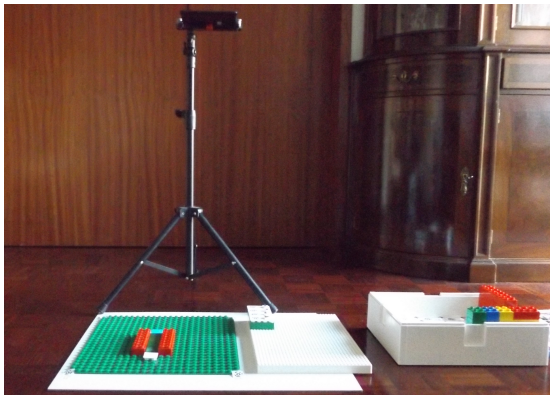

Figura 3.3: Protótipo de alta fidelidade.

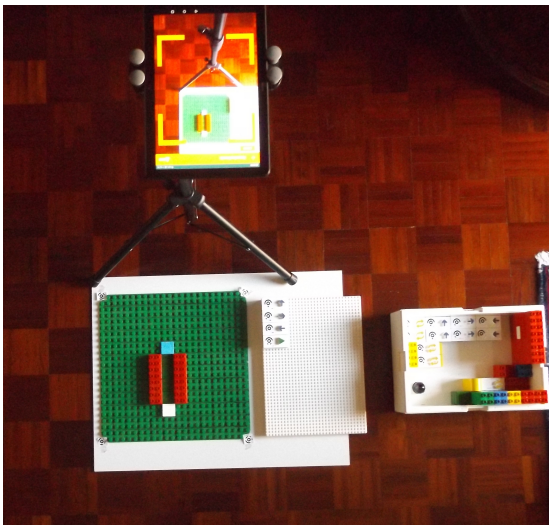

Figura 3.4: Protótipo de alta fidelidade (vista de cima).

3.4 Requisitos do Sistema

Qualquer sistema tem características a cumprir e estas são normalmente definidas através de requisitos, que no caso do sistema desenvolvido encontram-se relacionados com a interação específica, destinada a crianças cegas e com a simplicidade com que se entende o sistema.

Requisitos funcionais do sistema:

- **Reconhecimento do mapa (caminho)** - permite a recolha de informação por cor.
- **Reconhecimento dos tangíveis** - permite efetuar uma sequência através de blocos.
- **Reconhecimento de voz** - permite efetuar uma sequência através da resposta verbal a uma sequência de perguntas.
- **descarregamento da sequência para o robô** - Após a obtenção da sequência esta é transmitida para o robô.
- **Movimentação do robô** - execução do robô sobre o caminho construído.
- **Registo da pontuação** - permite registar em cada nível a pontuação e se já conseguiu ultrapassar o nível.

Requisitos não-funcionais do sistema:

- **Usabilidade** - aplicação com uma interface adaptada a utilizadores invisuais.

- **Personalizável** - Permite o reconhecimento de qualquer caminho efetuado pelo utilizador.
- **Acessível** - Permite a interação com o sistema, através do reconhecimento de voz e de tangíveis.
- **Instrutiva** - A aplicação indica todos os passos a efetuar.
- **Capacidade de aprendizagem** - O sistema possibilita a aprendizagem de diferentes conceitos de maneira progressiva pelos vários níveis.

3.5 Discussão

Neste capítulo, foram descritos cenários que ajudam a entender as várias interações que são permitidas pelo sistema, explicando a utilização de blocos tangíveis e do reconhecimento de voz, para a obtenção de uma sequência. O modo *freestyle* possibilita aos utilizadores construir caminhos arbitrários, criando assim novos níveis e o modo *accelerated* em que é necessário indicar a sequência de uma forma mais veloz.

Foram apresentados vários protótipos, com diferentes graus de fidelidade e níveis de detalhe, possibilitando explicar a evolução do sistema, ao longo de todas as fases até ao protótipo final. Estes protótipos permitem dar uma perspetiva de todo o processo e da visão generalizada do sistema, sendo demonstrados os diferentes componentes: o dispositivo necessário para a execução da aplicação, os blocos tangíveis e o mapa físico construído com *Legos*, que permite a deslocação do robô.

Entre os cenários e a prototipagem, foi possível construir uma ideia geral da composição do sistema e do seu funcionamento, de modo a perceber como estes se interligam.

Capítulo 4

Implementação

Neste capítulo, é apresentado o funcionamento do sistema desenvolvido para a aprendizagem do pensamento computacional, explicando detalhadamente a sua implementação e como se relacionam as diferentes componentes.

4.1 Visão Geral do Sistema

O sistema é composto por diversos componentes, encontrando-se estes divididos em: uma aplicação *Android* executada nos dispositivos *tablet* ou *smartphone* que são suportados por um tripé, possibilitando a interação com o sistema através de dois modos de jogo (modo por blocos e o modo por voz), numa plataforma que permite o encaixe de *Legos* com o objetivo de construir o mapa físico, numa base que permite o encaixe de blocos tangíveis de um modo sequencial, criando sequências e, por último, um robô designado de *Ozobot*. [11], que após a receção da sequência se movimenta no mapa construído.

4.2 Componentes Físicos

Nesta secção, são expostas todas as componente físicas referidas anteriormente e que compõem o sistema, elucidando todos os seus aspetos, características e a forma como se relacionam.

4.2.1 Mapa Físico

Um dos componentes é o mapa físico ou *board*, que consiste numa placa de cor verde, servindo de plataforma para a construção de um caminho, através de *Legos* de diferentes cores. Este mapa pode ser utilizado com dois objetivos: em primeiro lugar, permitir às crianças percecionar o caminho através do tato e, em segundo, permite a deslocação do robô *Ozobot [11]*.

Na figura 4.1 observa-se que o mapa se encontra dividido por traços pretos, para possibilitar uma melhor representação das diferentes casas da matriz, para a pessoa que for

construir o caminho, indicando onde é permitido o encaixe dos *Legos*.

Durante a fase de prototipagem foram escolhidas cores que fossem distintas para as diferentes casas que compõem o caminho, de modo a serem interpretadas pela aplicação. Por este motivo definiram-se as cores da seguinte forma:

- Branco indica a casa inicial do caminho.
- Verde representa o caminho disponível para se movimentar.
- Vermelho para os obstáculos ou "paredes"do caminho.
- Azul corresponde à casa final do caminho.

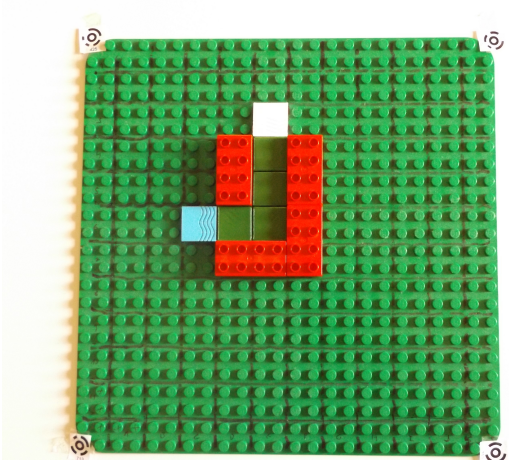

Figura 4.1: Exemplo de caminho construído com *Legos* no *board*.

Num sistema que é destinado à aprendizagem do pensamento computacional para pessoas cegas ou de baixa visão, é importante que as casas tenham cores diferentes, especialmente para pessoas de baixa visão, no entanto sem outro tipo de *feedback* que não este era impossível distinguir as diferentes casas do caminho. Para isso imprimiu-se numa impressora 3D, três *Legos* com texturas diferentes, tendo a casa branca que corresponde à casa inicial, uns traços mais retos e umas etiquetas adesivas redondas, a casa azul que corresponde à casa final, uns traços ondulados e a casa verde uma textura que é a normal do filamento usado, como é possível observar na figura 4.2. No caso dos obstáculos de cor vermelha, apresentados na figura 4.1 têm apenas a plataforma de encaixe como textura.

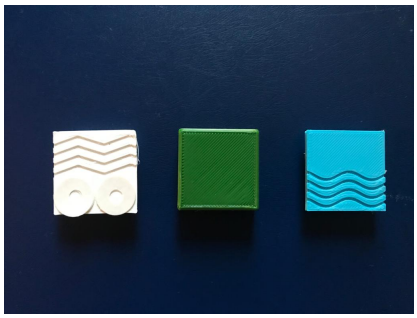

Figura 4.2: Textura das diferentes peças.

4.2.2 Blocos Tangíveis

Os blocos tangíveis são um dos componentes do sistema que permitem a criação de sequências de instruções, para posteriormente serem executadas pelo robô *Ozobot* [11]. Cada um destes blocos corresponde a uma instrução que realiza uma função diferente, considerando neste sistema as seguintes:

- Frente
- Esquerda
- Direita
- Início de ciclo
- Fim de ciclo
- *Play*

As instruções frente, esquerda e direita correspondem a blocos de movimento, sendo as suas respetivas funções, andar em frente, rodar 90º para a esquerda e rodar 90º para direita. As instruções de inicio e fim de ciclo, compõem o conceito de ciclo ou *loop* muito usado em programação, servindo para a repetição de instruções de movimento. O bloco de *play* indica ao sistema que a sequência se encontra pronta a ser executada.

Cada bloco contém um *design* específico, composto por um relevo que possibilita à criança cega, identificar a instrução a que corresponde e um código para que a aplicação a reconheça. Uma vez que a parte física do sistema é construída em torno de peças de *Lego*, estas foram aproveitadas para a criação de blocos, sendo que a parte dos relevos assenta sobre uma placa impressa em 3D, que pode ser encaixada no *lego* da cor desejada, permitindo assim que os blocos sejam colocados verticalmente numa placa branca, que serve de plataforma para a construção do programa.

Os blocos de movimento são representados com setas para as suas respetivas direções (figura 4.3 - com setas azuis) e os blocos de ciclo com duas setas, dando a ideia de uma

repetição. Estes últimos contém ainda *legos* do lado esquerdo que simulam o número de repetições, ou seja, se tiver dois *legos*, significa, que repete todas as instruções duas vezes (figura 4.6). O bloco de *play* apresenta o triângulo que é associado à função que se pretende iniciar (figura 4.3 - último bloco a verde).

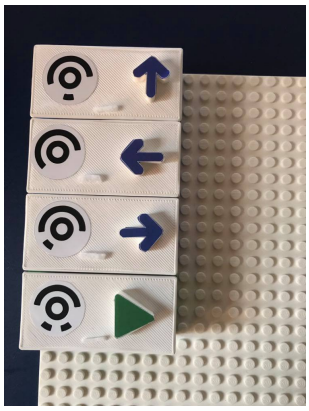

Figura 4.3: Exemplo de programa com tangíveis.

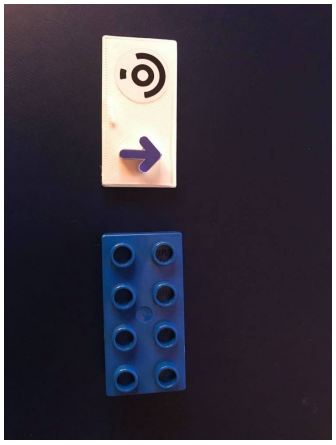

Figura 4.4: composição do bloco (parte da frente).

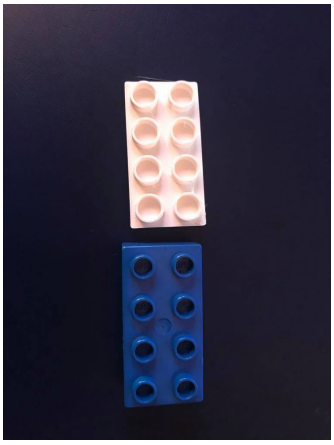

Figura 4.5: composição do bloco (parte de trás).

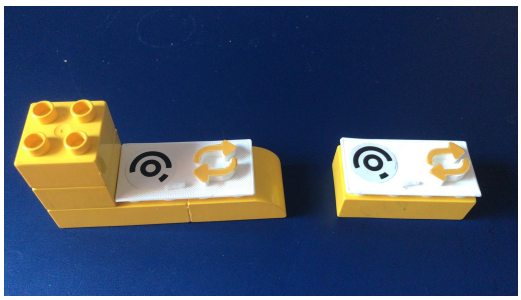

Figura 4.6: Bloco de início (esquerda) e fim de loop (direita) .

4.2.3 Robô *Ozobot*

O robô que foi escolhido para o sistema é designado por *ozobot* [11] (ver figura 4.7). Este é um robô que possui uma característica fundamental para que o sistema funcione, tendo dimensões muito pequenas, tornando-o ágil para se deslocar no caminho construído com *legos* do mapa físico.

Este robô tem a particularidade de receber as instruções por uma sequência de cores, que vão aparecendo num ecrã, descodificando-as através do sensor (ver figura 4.8), que se encontra na zona inferior junto às suas rodas, obtendo assim as instruções das ações que deve executar.

Figura 4.7: Robô *Ozobot* [11].

Figura 4.8: sensor do robô *Ozobot* [11].

4.2.4 *Tablet e Tripé*

Neste sistema, uma funcionalidade essencial é o reconhecimento do *board*, que só é possível através do desenvolvimento de uma aplicação *Android*, sendo necessário ter um dispositivo *tablet* ou *smartphone*, que se coloca de forma paralela a este e de modo a que esteja dentro do campo de visão da câmara.

Para suportar o dispositivo foi adquirido um tripé, como o que se encontra na figura 4.9, em que a altura é ajustável e a base que segura o dispositivo não interfere com o posicionamento da câmara.

Figura 4.9: Tripé a suportar o *tablet*.

4.3 Arquitetura do Sistema

Nesta secção é apresentada uma visão geral da arquitetura da aplicação do sistema, que se encontra dividida em 4 partes interligadas entre si, configurando assim a interface do utilizador, lógica da aplicação, base de dados e uma *python framework* designada *Chaquopy* [2].

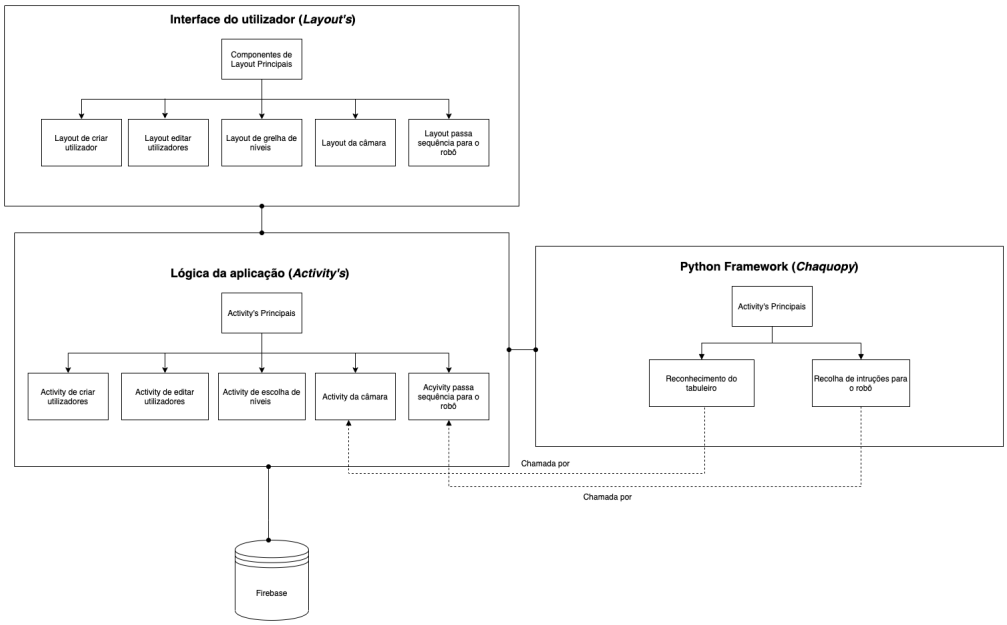

Figura 4.10: Arquitetura da aplicação.

A interface do utilizador funciona como uma camada de apresentação da aplicação, sendo composta por todos os ficheiros de *layout* que incluem a programação na linguagem *XML*, sendo responsáveis por todos os elementos de apresentação visual.

A lógica da aplicação funciona como a camada de negócio da aplicação, sendo constituída por todos os ficheiros *Java*, também designados de *Activity's*. Estas são responsáveis pela execução das funções dos elementos da interface do utilizador e de os interligar com a lógica que é executada em *background*, necessária para que o sistema funcione corretamente.

A *Python framework* (*Chaquopy* [2]) é uma ferramenta que tem a capacidade de fornecer um ambiente de execução da linguagem *Python* numa aplicação que está preparada para executar a linguagem *Java*. No contexto do sistema esta *framework* é responsável por executar toda a lógica que foi desenvolvida na linguagem *Python*, ou seja, pelo algoritmo que divide a imagem em cada casa da matriz e por transformar as instruções indicadas pelos utilizadores, numa sequência de cores, possibilitando posteriormente a sua leitura, através do sensor do robô.

A base de dados corresponde à camada de dados, na qual são guardadas informações relativas aos utilizadores. Neste sistema são inseridos os utilizadores, sendo estes com-

postos por uma *tag*, que referencia a pessoa e à qual se encontra associado o número de pontos, que o utilizador adquiriu em cada nível e uma lista com os níveis que o mesmo ultrapassou.

4.4 Interface do Utilizador

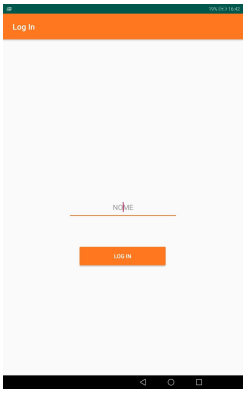

Figura 4.11: Ecrã Inicial.

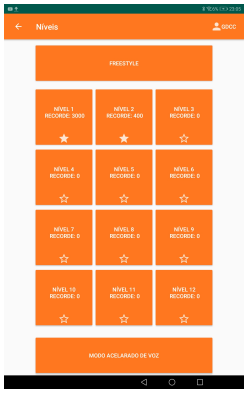

Figura 4.12: Ecrã dos níveis.

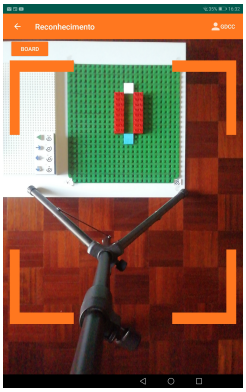

Figura 4.13: Ecrã da câmara.

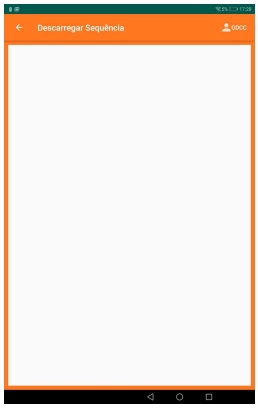

Figura 4.14: Ecrã de transmissão da sequência para o robô.

A interface da aplicação tem como objetivo permitir que o jogo seja realizado de uma forma intuitiva e que crianças invisuais possam interagir com o sistema, após a instalação de todo o ambiente indispensável para a realização da atividade, sem ajuda de outros intervenientes.

O ecrã inicial (figura 4.11) é uma interface que permite que seja inserida uma *tag* correspondente a um utilizador e apenas cria um novo, se a *tag* ainda não existir e após esta autenticação, a aplicação segue para o ecrã dos níveis.

O ecrã dos níveis (figura 4.12) é uma interface que contém um conjunto de botões que permitem não só escolher o nível que desejam começar a jogar, tal como percecionar os

pontos que obtiveram em cada nível e se terminaram ou não os níveis com sucesso. Em cada botão é possível verificar esta informação, pois mostram os pontos e uma estrela que se apresenta preenchida no caso de o nível ter sido ultrapassado e não preenchida no caso contrário. Depois de selecionar um dos níveis, a aplicação encaminha o utilizador para o ecrã da câmara.

Na interface da câmara (figura 4.13) é a interface que possibilita o início do jogo, ou seja, nesta vão ser dadas instruções verbais, sobre a forma como começar o nível e através da tecnologia de toques, é possível iniciar o modo conversacional, onde são efetuadas várias questões à criança, para esta indicar o programa que quer executar, sendo também possível iniciar o modo de programação por blocos. Após a iniciação do nível pelos diferentes toques possíveis, a aplicação capta uma fotografia, para possibilitar o reconhecimento do *board*, avançando para a recolha da sequência por blocos ou por voz. Quando terminado o reconhecimento do *board* e a recolha da sequência, a aplicação passa para o ecrã de descarga da sequência.

O ecrã de carregamento da sequência para o robô (figura 4.14), consiste numa interface que apresenta um quadrado branco com uma *frame* cor de laranja que ocupa quase toda a tela do dispositivo, sendo esta, a área disponível para colocar o robô. Esta área ocupa um grande espaço, para que uma criança cega tenha liberdade de colocação do robô, uma vez que esta tem de posicionar o sensor do robô de forma a conseguir que este decifre a sequência de cores e se transmitam as ações a realizar.

4.5 Modos de Interação da Aplicação

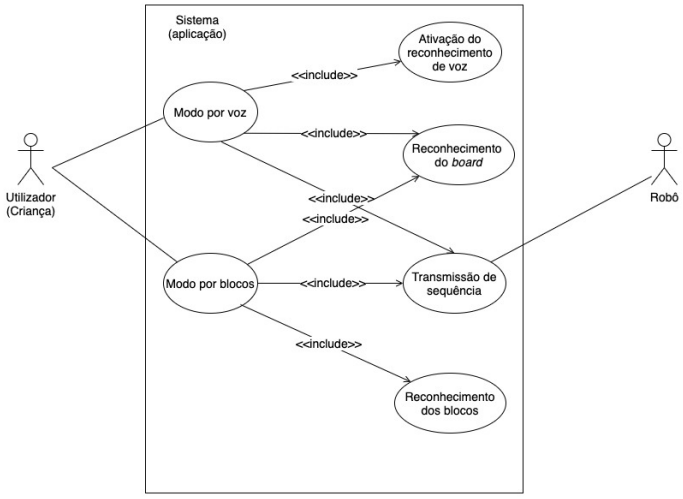

Figura 4.15: Diagrama de casos de uso.

Nesta secção são demonstrados os diferentes modos de interação que o sistema permite. Como é possível observar na figura 4.15, existem dois modos : o modo por voz e o modo por blocos.

O modo por voz consiste numa interação verbal com o sistema, na qual possibilita a obtenção da sequência a ser executada pelo robô. Esta interação ocorre durante uma série de questões que são efetuadas e as respostas dadas, são traduzidas, na sequência desejada pelo utilizador. Na figura 4.16 é possível observar um exemplo de interação do nível 1.

Cada nível é diferente e, por esse motivo, as questões também vão variando, para que a aprendizagem da criança seja progressiva. A imagem 4.16 representa o diálogo que é realizado entre a criança e o assistente da aplicação designado de *Botnik*.

NIVEL 1
BOTNIK: EU SOU O BOTNIK A TORRE DE CONTROLO DO ROBÔ! O ROBÔ PRECISA DA TUA AJUDA!
BOTNIK: ESTOU NO PLANETA NABIDA PRECISO DE SAIR DESTE AUTENTICO DESERTO! AJUDA-ME!
BOTNIK: ELE NÃO CONSEGUE ANDAR SOZINHO SEM AS MINHAS INSTRUÇÕES! MAS EU NÃO CONSIGO VER O CAMINHO PARA LHE DIZER AO ROBÔ!
BOTNIK: PRECISO DA TUA AJUDA PARA DIZER O CAMINHO AO ROBÔ! QUERES AJUDAR?
BOTNIK: ACHAS QUE DEVIA IR PARA A FRENTE OU VIRAR PARA A DIREITA OU VIRAR A ESQUERDA?
CRIANÇA: NÃO SEI!
BOTNIK: TENS DE ESCOLHER!
BOTNIK: ACHAS QUE DEVA IR PARA A FRENTE OU VIRAR PARA A DIREITA OU VIRAR A ESQUERDA?
CRIANÇA: FRENTE!
BOTNIK: RECEBIDO A INSTRUÇÃO FRENTE! MAIS ALGUMA INSTRUÇÃO?
CRIANÇA: NÃO! (NO CASO "SIM" VOLTA PARA A LINHA 21)
BOTNIK: SE JA TERMINASTE DIZ TERMINAR!
CRIANÇA: TERMINAR!
BOTNIK: JÁ TENHO AS INSTRUÇÕES! AGORA PRECISO DE COMUNICAR AO ROBÔ!
BOTNIK:PRECISO QUE PEGUES NO ROBÔ E COLOQUES NO TABLET!
BOTNIK:PEGA NO ROBÔ E COLOCA-O NA CASA DE PARTIDA!
BOTNIK: BOA CHEGASTE AO FIM!
BOTNIK: SE QUISERES VOLTA A REPETIR TUDO O QUE FIZESTE ANTES! CARREGA DURANTE SEGUNDOS NO ECRÃ, PARA DARES NOVAS INSTRUÇÕES AO ROBÔ!

Figura 4.16: Exemplo de interação por voz (Nível 1).

O modo por blocos consiste numa interação com o sistema, através de peças tangíveis que permitem definir a sequência a executar pelo robô. Estas contêm um relevo, que possibilita a crianças invisuais identificar a instrução e um código *Topcode* [27] que, por sua vez, é reconhecido pelo sistema.

Estes blocos são colocados sobre uma placa branca onde o seu encaixe é efetuado na vertical e, quando este modo é iniciado, existe um *timer* [15] que continua a capturar fotografias, enquanto a peça de *play* não estiver presente. Assim que este bloco for reconhecido pela aplicação, obtém-se a sequência total desenvolvida pelo utilizador.

4.6 Reconhecimento do Caminho Construído com *Legos*

Nesta secção, descreve-se como é realizado o reconhecimento do caminho de *legos* mais detalhadamente e ao nível da implementação.

Para o *board* de *legos* decidiu-se colocar códigos *Topcodes* [27] nos seus quatro cantos, de modo a ser possível identificar as suas coordenadas e desta forma saber os seus limites.

O reconhecimento do *board* só é possível ser feito através da análise de imagens,

por esse motivo, através da câmara do dispositivo conseguimos capturar a fotografia que contém o caminho. Posteriormente, a fotografia é transformada num objeto *Java Bitmap* [1] e a partir deste é realizado um *scan*, possibilitando o reconhecimento dos códigos *TopCode* [27], presentes nos quatro cantos referidos anteriormente, de forma a obter as coordenadas de cada um deles (considera-se um referencial bidimensional). Esta função *scan* é chamada do Objeto *Java Scanner* da API do *TopCode* [27], que devolve um objeto *List* [8] contendo todos os códigos encontrados na imagem.

Partindo das coordenadas obtidas, foi realizado um algoritmo que permite calcular todos os pontos das diferentes casas da matriz, pressupondo que o *board* disponível tem tamanho 12 x 12. Com estes pontos, a fotografia é recortada, permitindo isolar cada casa, analisando uma de cada vez.

Esta análise é feita em cada *píxel* da imagem através dos valores de RGB, ou seja, são acumulados os valores de vermelho, verde e azul em variáveis diferentes e, em seguida, é verificado qual é o maior valor, sendo este o que determinada a cor de cada casa. Na casa branca é avaliada a condição da imagem, se esta contém valores de verde, vermelho e azul, igual ou superior a um *threshold* de 150 (ver figura 4.17).

```
for (int y = 0; y < bmp2.getHeight(); y++) {
    for (int x = 0; x < bmp2.getWidth(); x++) {
        int color = bmp2.getPixel(x, y);
        redColors = Color.red(color);
        greenColors = Color.green(color);
        blueColors = Color.blue(color);

        if (greenColors >= 150 && redColors >= 150 && blueColors >= 150) {
            white++;
        } else if ((greenColors >= redColors && greenColors >= blueColors)) {
            green++;
        } else if (redColors >= greenColors && redColors >= blueColors) {
            red++;
        } else if (blueColors >= redColors && blueColors >= greenColors) {
            blue++;
        }


    }
}
```

Figura 4.17: parte do código da análise da cor.

Em seguida, consoante a cor obtida, é efetuado uma conversão do *board* para uma notação, que permite mais tarde realizar a análise do caminho do robô. A cor branca é atribuída ao início do percurso, o vermelho para os obstáculos, verde para indicar o caminho que se encontra livre e o azul para o final.

4.7 Reconhecimento de Blocos Tangíveis

Nesta secção é descrito como é realizado o reconhecimento de blocos tangíveis de forma mais detalhada e ao nível da implementação.

Os blocos tangíveis contêm um código *TopCode* [27] que permite a sua identificação por parte da aplicação, sendo que a cada código está atribuída uma função específica.

O reconhecimento dos blocos tangíveis é feito a partir de uma fotografia, que é capturada com a câmara do dispositivo. Esta foto é transformada num objeto *Java Bitmap* [1] que a partir do qual, é realizado um *scan*, possibilitando o reconhecimento dos códigos *TopCode* [27] presentes em cada bloco, de forma a obter as coordenadas de cada um (considera-se um referencial bidimensional).

Em seguida, é realizado um algoritmo a partir das coordenadas obtidas, ordena-as pelo eixo das ordenadas, uma vez que estes se encontram dispostos na vertical (de cima para baixo).Depois dos códigos terem sido ordenados é feita uma transformação do código do *TopCode* [27], para uma notação num objeto *Java String* [13], permitindo posteriormente a análise da movimentação possível do robô no caminho efetuado.

4.8 Reconhecimento de Voz

Na secção dos modos de interação, foi explicado o que acontece, de um modo geral, quando o modo por voz é ativado e nesta secção é explicado de uma forma mais detalhada como se encontra implementado.

Quando se inicia este modo, a aplicação fornece uma história diferente em cada nível, tendo sido programado através de um objeto *Java Timer* [15] que permite temporizar as falas que são acionadas por um objeto *Java TextToSpeech* [14], que é responsável por converter texto em fala.

Após a introdução do nível, segue-se a recolha da sequência por uma estratégia conversacional, que é iniciada através do método *lauchSpeechRecognition()*. Este método é responsável por abrir a janela do reconhecimento de voz que faz uso da API produzido pela *Google*, efetuando este todo o processamento de tradução da voz em texto.

A partir do objeto *Java String* [13] resultante, é realizada uma análise para que se possa converter as instruções numa notação mais conveniente e de modo a que seja possível despoletar a fala seguinte.

4.9 Execução do Robô

Após a definição da sequência de instruções a ser executada pelo utilizador, aparece na aplicação uma interface que contém um quadrado branco, que corresponde ao ecrã da descarga da sequência, referido anteriormente, servindo assim de plataforma para a passagem da sequência de instruções para o robô. Quando lançada esta página, são escritas num ficheiro as várias instruções que permitem a movimentação do robô.

Depois é realizada uma chamada a um objeto, que permite a execução de ficheiros existentes na *Python Framework chaquopy* [2] responsáveis pela tradução das instruções,

para um *Objeto String* [13], que permite posteriormente alterar o *background* da página. Este é modificado através de um objeto *Java Timer* [15] que a cada 50 milissegundos, obtém a cor do método *getColor()*, permitindo assim a passagem da sequência para o robô.

Posteriormente é essencial calibrar o robô, efetuando dois passos antes de iniciar a sequência de cores. O primeiro passo é calibrar o robô com a superfície branca, sendo necessário carregar no botão disponível e esperar que este apresente uma luz verde, indicando assim que se encontra pronto para receber a sequência. No segundo passo, volta-se a carregar no botão e a colocar o robô no *tablet* e assim que for iniciada a sequência, o sensor do robô capta as várias cores, interpretando as diferentes instruções. No final, se o robô apresentar a cor verde, a passagem foi concluída com sucesso, se tal não acontecer, apresentará cor vermelha tal como ilustrado na figura 4.18.

Figura 4.18: Cores de calibração do *Ozobot* [11].

4.10 Armazenamento de Dados

Nesta secção é descrito como é realizado o armazenamento de dados de forma mais detalhada e ao nível da implementação.

Os dados são armazenados numa base de dados designada *Firebase* [26], sendo apenas guardadas informações relativas a utilizadores, contendo estes uma lista com os pontos obtidos em cada nível e uma outra lista que indica os níveis que foram concluídos. Na figura 4.19 é possível observar a hierarquia de informações armazenadas, contendo em primeiro lugar a referência dos utilizadores (*Users*), por baixo hierarquicamente encontra-se a *tag* com as iniciais do nome do utilizador correspondente, sendo este composto pelas duas listas anteriormente referidas.

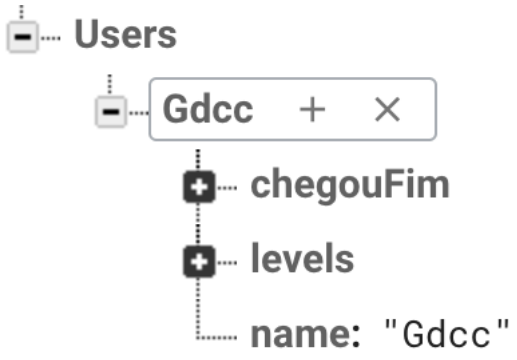

Figura 4.19: Esquema de utilizadores do *firebase* [26].

A introdução e a obtenção de valores da base de dados é efetuada a partir de *Listeners*, que estão sempre à espera de um determinado evento para introduzirem ou obter os dados necessários.

4.11 Ferramentas Utilizadas

Nesta secção, são abordadas as várias tecnologias e ferramentas que permitiram a concretização do sistema.

Para delimitar o *board* de *Legos* e definir as instruções dos blocos tangíveis, foram usados códigos *Topcode* [27], que são reconhecidos através de funções fornecidas pela sua API e utilizadas sobre a imagem captada pela aplicação.

O reconhecimento das cores do *board* de *Legos* foi efetuado com recurso à biblioteca *OpenCV* [41], que permite o recorte da matriz da imagem inicial nas diferentes casas e uma análise da cor de cada pixel, assumindo em cada casa a cor mais dominante.

A aplicação foi realizada na linguagem *Android*, com a maior parte da lógica desenvolvida em *Java* e as interfaces em *XML*.

Para se conseguir movimentar o robô foi necessário a execução de um projeto, através da *framework Chaquopy* integrada na aplicação *Android*. Este encontrava-se num repositório do *GitHub* [32], que convertia instruções em *Python* para a linguagem máquina do robô [2].

O Armazenamento de informações de utilizadores, pontos e níveis terminados, só foi possível com o serviço de base de dados *Firebase* fornecido pela *Google*.

Capítulo 5

Avaliação Preliminar

Neste capítulo, é mostrada a avaliação realizada, com base no sistema desenvolvido e são apresentados os resultados dos inquéritos respondidos por educadores. O objetivo desta avaliação é entender a adequação do sistema no desenvolvimento do pensamento computacional, assim como obter sugestões sobre as vantagens e limitações do mesmo.

Durante o processo de avaliação, é necessário salientar que nos encontrávamos em plena pandemia do coronavírus (COVID-19) e, por esse motivo, tornou-se impossível realizar uma avaliação presencial, pelo facto de existirem limitações nas escolas e instituições que apoiam pessoas cegas e de baixa visão, não havendo possibilidade de pais e educadores de colaborarem no estudo com o protótipo, durante este período. Perante esta situação, decidimos realizar um vídeo, no qual é demonstrado o funcionamento do sistema, explicando verbalmente todas as funcionalidades.

Em seguida, elaborou-se um inquérito no qual são realizadas questões gerais e específicas sobre a utilização do sistema e cada uma das suas componentes e outras questões que permitem a caracterização da população que respondeu ao inquérito (ver em anexo Questionário). Neste deu-se a possibilidade de os inquiridos responderem verbalmente ou através de texto, às perguntas de carácter mais extenso.

Após o processo relatado nos parágrafos anteriores e a obtenção de resultados dos inquéritos, procedeu-se à sua análise, levando-nos a uma reflexão sobre as sugestões que consideramos mais relevantes e que serão expostas nas secções seguintes.

5.1 Apresentação do Conceito e Ferramenta

Nesta secção, é abordado todo o processo realizado na demonstração das funcionalidades do sistema num vídeo, assim como o seu propósito e necessidade.

O facto de as pessoas não conseguirem ter acesso ao sistema, de forma a poderem testá-lo, surgiu a alternativa de produzir um vídeo que demonstrasse o seu funcionamento e todas as suas componentes.

Seguidamente, selecionaram-se as componentes importantes que focam o objetivo do

sistema para permitir a conceção do questionário. Depois procedeu-se à construção do vídeo pelo *board* de *Legos*, a fim de mostrar a possibilidade de montagem de qualquer tipo de caminho com peças *Lego* e a correspondência das suas diferentes cores, com o intuito de questionar se era uma solução de simples construção, viável e lúdica para as crianças.

Posteriormente, passou-se ao cenário de utilização de tangíveis, expondo as várias peças a serem encaixadas, originando uma sequência de instruções de maneira a que o espectador se inteirasse desta forma de programar e pudesse efetuar questões sobre a utilidade e interação com os tangíveis.

Em seguida, apresentou-se a interação com o modo de voz, para demonstrar a existência deste modo inovador de programar sequências e poder questionar, para em seguida se apurar qual a faixa etária mais ajustada a este modo e se seria uma boa solução em ambiente escolar ou em casa.

Por último, surge o robô em movimento a executar as sequências criadas no modo por blocos e no modo por voz, com o objetivo de questionar acerca do impacto causado pelo tamanho do robô e se apresenta um *feedback* explícito.

Para finalizar, na edição do vídeo foram colocados áudios explicando cada componente, enquanto estas estão a ser visualizadas.

Este vídeo foi publicado no *website* do *youtube* [21], para permitir que a plataforma do questionário (jotform [6]) possibilitasse a sua divulgação (ver anexo Questionário). Esta plataforma foi escolhida pelo facto de ser a única que proporciona a resposta por gravação de voz ou texto.

5.2 Avaliação com Educadores

5.2.1 Objetivos

De uma forma geral, os inquéritos serviram para obter *feedback* por parte de pais, educadores e investigadores, sobre a adequação do sistema à aprendizagem do pensamento computacional em contextos diferentes (em casa e em sala de aula) e sugestões que pudessem melhorar o sistema.

Nesta forma de avaliar, consideraram-se como objetivos:

- Perceber para que faixas etárias o sistema é mais adequado, segundo a opinião e experiência dos participantes.
- Recolher vantagens e limitações do sistema.
- Avaliar as possibilidades do sistema em diferentes ambientes (em casa ou em sala de aula).
- Obter sugestões de melhorias que possam ser efetuadas ao sistema.

- Obter sugestões sobre detalhes de cada componente, de forma mais específica.

5.2.2 Procedimento

Após a conclusão do protótipo, procedeu-se à fase de avaliação, com o intuito de entender quais as suas possibilidades e limitações. Devido às restrições anteriormente referidas, foi necessário encaminhar o estudo noutra direção e, nessa sequência, elaborou-se um inquérito (ver em anexo Questionário) com o formato disponibilizado pela plataforma *jotform* [6], por ser esta a única que permitia a utilização de *input* de voz, possibilitando ainda a sua partilha de um modo mais fácil e rápido, através de um link gerado por esta plataforma. Este inquérito é efetuado com base num vídeo, que demonstra a utilização do sistema.

Em seguida, para se conseguir algumas respostas, de forma a obter *feedback*, divulgou-se os inquéritos para escolas e instituições de apoio a pessoas cegas e de baixa visão, via e-mail, no qual foi enviado um link que ao clicar, direcionava para o questionário realizado na plataforma *jotform* [6].

Ao abrir o questionário, aparece um consentimento informado que relata que o inquérito é anónimo e submetido de forma voluntária. Na página seguinte descreve-se o sistema e é possível visualizar o vídeo que o demonstra. Este mesmo questionário dispõe de 9 páginas e cada uma delas contém uma secção que corresponde a uma componente, com múltiplas questões cujo o número varia num intervalo de 3 a 7, somando um total de 20 perguntas, das quais apenas 3 não foram respondidas.

Na última página, questionava-se o inquirido se desejava usar o sistema, podendo escrever o seu *e-mail* para um futuro contacto.

Posteriormente, procedeu-se à análise das respostas dos respetivos inquéritos de modo a poder expor os resultados, que se encontram descritos na última secção deste capítulo.

5.2.3 Inquéritos

Os inquéritos foram efetuados a pais, educadores e investigadores, em que apenas um educador de necessidades educativas especiais, já tinha participado em estudos de outros projetos, no âmbito da aprendizagem do pensamento computacional.

Nas secções seguintes, serão abordadas as características da amostra populacional que respondeu aos inquéritos e analisados os resultados obtidos, destacando as sugestões mais recomendadas e que façam sentido no contexto do projeto.

5.2.4 Participantes

Nesta secção, é efetuada a caracterização dos quatro participantes que responderam ao inquérito:

O primeiro inquirido é um educador de tecnologias de informação/robótica, que ensina crianças na faixa etária entre os 7 e os 14 anos, mas nunca trabalhou com crianças com deficiências visuais.

O segundo inquirido é um educador, que já deu apoio a crianças com necessidades educativas especiais com idades entre os 7 e os 14 anos de idade, não tendo especificado o tempo de serviço.

O terceiro inquirido é um educador de tecnologias de informação/robótica, que nunca trabalhou com crianças com deficiências visuais, mas leciona crianças com idades compreendidas entre os 7 e os 14 anos.

O quarto inquirido é um educador que apoia crianças com necessidades educativas especiais, entre os 10 e os 12 anos de idade e tem 34 anos de experiência profissional.

5.2.5 Discussão

Após a obtenção de várias respostas aos inquéritos, foi possível recolher informações importantes acerca do que os inquiridos pensam sobre as qualidades e limitações do sistema e analisar algumas sugestões que possibilitam melhorar o sistema.

5.2.5.1 Faixa Etária Adequada ao Sistema

Relativamente à faixa etária que os inquiridos consideram ser a ideal para o uso do sistema, é comum a todas as respostas, o intervalo compreendido entre os 7 e os 12 anos, justificando que nestas idades, as crianças já têm as capacidades de lógica, compreensão e retenção de conceitos suficientemente desenvolvidos para conseguirem utilizar o sistema de forma autónoma, como mencionado por um dos inquiridos:

"Normalmente já são crianças do 4º ano para cima, pelo que já entendem melhor o funcionamento do jogo e a lógica de programação."

5.2.5.2 Qualidades do Sistema

Nas qualidades mais apreciadas do sistema, os inquiridos entenderam que este poderá beneficiar o desenvolvimento cognitivo e mental da criança, assim como a memória, o raciocínio, a lateralidade e orientação, como referido pela citação que se segue:

"Permite estimular capacidades cognitivas como a memória, raciocínio. Permite também desenvolver a lateralidade e a orientação."

O desenvolvimento do pensamento abstrato está relacionado com a forma como as crianças associam e interpretam conceitos, podendo levar a uma maior evolução do seu raciocínio, assim como lidar melhor com práticas recorrentes na construção de algoritmos,

como a abstração e decomposição de tarefas que podem ter benefícios nas ações do seu quotidiano [44, 50].

A perceção do *board* de *legos* e a movimentação do robô permite que a criança indique mudanças de direção, possibilitando um desenvolvimento na sua lateralidade e orientação espacial que, por sua vez, integra a aprendizagem percetual e cognitiva [44].

Os inquiridos salientam também a facilidade de uso do sistema, como se pode ler na seguinte resposta:

"A gravação da voz e a verificação das instruções funciona bastante bem. A utilização de legos que, não só já são conhecidos pela grande maioria das crianças, mas também constituem uma ótima solução para crianças com perda de visão ou invisuais."

5.2.5.3 Limitações do Sistema

Quanto às limitações do sistema, apontam como uma possível falha, o ambiente caótico que se vive numa sala de aula, podendo não proporcionar um bom reconhecimento da voz e sistemas devido ao barulho existente. Referem também que existe uma temporização longa da recolha de instruções, no modo de voz, como se pode observar pelas seguintes respostas:

"Para trabalhar em sala de aula pode ser mais complicado devido ao cruzamento de sons."

"O sistema poderia ser mais rápido de processar a informação, bem como a verificação das instruções."

5.2.5.4 Disciplinas Adequadas ao Uso do Sistema

Os inquiridos indicam que o sistema possibilita o treino de capacidades como desenvolvimento psicomotor e consideram que pode ser usado em disciplinas como a Matemática e TIC (tecnologia da informação e comunicação), referindo a importância de jogos didáticos com os sistemas, como corridas em modo colaborativo. Esta análise é corroborada pelas seguintes respostas:

"Adequado ao desenvolvimento psicomotor"

O desenvolvimento psicomotor refere-se a mudanças nas habilidades cognitivas, emocionais, motoras e sociais de uma criança, desde o início da vida até à adolescência [16].

Para o desenvolvimento de habilidades motoras essenciais, o período mais importante é o da primeira infância, na verdade, já na pré-escola as crianças são naturalmente curiosas e dispostas a explorar especialmente por meio de brincadeiras ou adequadamente

estimuladas a realizar uma atividade física [16].

A intencionalidade está vinculada à capacidade de uma criança conseguir ligar a ação que efetua a colocação de um bloco tangível, provocando o movimento do robô, por exemplo cria essa ligação ao colocar o bloco que corresponde à instrução frente e este caminha nessa direção, concluindo assim a respetiva ação.

"Matemática, TIC, Lógica e resolução de problemas."

Há uma espécie de experiência lógico-matemática que surge quando a criança constrói relações entre os objetos, ou melhor, entre as suas ações sobre os objetos [24]. Por exemplo, caso a criança realize uma contagem das casas do caminho construído com *legos*, poderá fazê-la em ambos os sentidos (de cima para baixo e de baixo para cima), existindo uma relação entre os *legos*, pelo que ela conseguiu uma organização das suas ações relativamente a estes. Conclui-se, assim, que pode ser aplicado ao nível da lógica e da matemática.

5.2.5.5 Ambiente Adequado à Utilização do Sistema

A maior parte das respostas acerca da utilização em ambiente da sala de aula ou em casa, são positivas no contexto de casa, salientando que pode ser uma atividade lúdica e até terapêutica. No contexto de sala de aula, as respostas são negativas, referindo que o sistema nesse ambiente seria desafiante mas demasiado barulhento, como se pode verificar pelas afirmações que se seguem:

"Cada criança poderia ter o seu próprio tabuleiro e peças lego e podiam ser feitos jogos/exercícios de corridas do robot, enigmas, ou labirintos que as crianças tivessem de resolver utilizando as peças como instruções."

5.2.5.6 Componentes do Sistema

Entrando nas questões mais específicas dos componentes, quanto ao mapa físico, os inquiridos aludem que este trabalha a sensibilidade, orientação e memória, podendo servir como consulta durante a atividade, fácil de montar e lúdico, como se pode observar na afirmação seguinte:

"Simples, as ordens mantém-se percetíveis, a sequência permanece à disposição e pode ser consultada."

Acrescentam ainda, que esta componente pode servir para a aprendizagem de pequenos trajetos que a criança tem de realizar no seu dia a dia.

"Talvez possam ser simulados trajetos que a criança precise aprender no seu dia-a-dia. Pequenos percursos apenas."

Um dos inquiridos referiu que nesta componente, o tamanho do *board* tem uma dimensão reduzida, mas que deveria ser maior, para possibilitar a criação de caminhos mais longos e complexos, como demonstra a citação que segue:

"Deveria ser possível juntar mais que um tabuleiro para aumentar o mapa."

Na componente dos blocos tangíveis, avaliam como uma solução lúdica, e, por esse motivo, desperta mais o interesse e curiosidade da criança, considerando que se encontram desenhados de forma simples e que podem possibilitar uma identificação imediata.

"As peças de lego estão sempre associadas a algo lúdico, logo pode ser mais atrativo para despertar o interessa à criança"

Na componente que permite a interação por voz, os inquiridos responderam que este tipo de interação deve ser direcionado a faixas etárias mais jovens e consideram uma boa forma de comunicação para pessoas cegas, pois permite que se foque na atividade que está a realizar, como explicita a afirmação:

"A voz é sempre uma boa forma de comunicação para um cego, pois faz com que esteja mais concentrado e focado na atividade."

Uma das soluções indicadas foi o facto de que poderiam ser colocados uns *phones* com tecnologia *Bluetooth* para o contexto de sala de aula e que a interação deveria ser mais rápida, como se verifica na citação abaixo indicada:

"Poderia ser interessante associar uns phones por bluetooth, por exemplo, para contexto de sala de aula"

Numa das resposta foi exposto que a interação por voz apresentava uma velocidade lenta na obtenção das sequências, como referido na seguinte:

"A verificação das instruções poderia ser mais rápida, ou feita a cada x instruções. Deveria ser possível fazer várias instruções de uma só vez "

No caso da componente do robô, apenas indicam que este parece ser ideal e que não é necessário outro tipo de *feedback* para além daquele que é transmitido pelo robô e pela aplicação, sendo o suficiente para não comprometer o entendimento da atividade, como se confirma pela afirmação subsequente:

"Penso que o feedback dado é claro e que está bem realizado."

5.2.5.7 Resumo

Em suma, apesar das limitações de concretização de uma avaliação presencial, foi possível adapta-la através de um inquérito, em que foram efetuadas questões sobre todas as componentes do sistema com base na visualização de um vídeo, de modo a obter *feedback* de possíveis melhorarias a realizar no sistema.

Nas respostas aos inquéritos, verificámos que o sistema tem características muito positivas, mas também algumas limitações que podem ser melhoradas. Com as sugestões dos inquiridos, encontrámos novos caminhos para tornar o sistema mais prático, eficiente e mais acessível para os utilizadores.

Capítulo 6

Conclusões

Nos nossos dias, cada vez existe mais uma dependência de dispositivos que nos permitem melhorar a execução de tarefas no âmbito profissional e pessoal, permitindo-nos organizar no nosso quotidiano. O desenvolvimento do pensamento computacional pode permitir melhorar a forma, como realizarmos tarefas para as quais existem um conjunto de passos, que são necessários efetuar.

Atualmente, há estudos que indicam que a aprendizagem do pensamento computacional em idades precoces têm benefícios a longo prazo, influenciando o futuro pessoal e profissional dos indivíduos.

No entanto, ainda existem alguns problemas de acessibilidade na obtenção de conhecimentos para pessoas cegas e de baixa visão, nomeadamente ao nível da aprendizagem de conceitos de programação que permitem um desenvolvimento do pensamento computacional. Esta fraca acessibilidade traz limitações que podem ter influência nas escolhas futuras, pois não permitem adquirir as competências necessárias no âmbito da engenharia.

No entanto, têm-se reunido esforços para atenuar estas dificuldades, através do desenvolvimento de sistemas que introduzem conceitos de programação, a crianças cegas com interfaces gráficas, de voz ou tangíveis, possibilitando uma evolução ao nível do pensamento computacional e das capacidades cognitivas da criança, levando-a a lidar melhor com práticas recorrentes na construção de algoritmos, como a abstração ou decomposição de tarefas.

De forma a atenuar as dificuldades anteriormente referidas e com o objetivo de permitir que crianças tenham acesso facilitado a conceitos de programação, desenvolveu-se um sistema que contempla uma aplicação *Android*, que pode ser executada num dispositivo *tablet* ou *smartphone*. Esta é responsável pelo reconhecimento de um caminho por cores, construído com *legos*, sobre uma plataforma de encaixe. Este sistema possui a capacidade de criar sequências, por uma interação com blocos ou através do reconhecimento de voz, com o intuito de transmitir essa sequência a um robô, que se desloca sobre o mapa de *legos*. O sistema tem a particularidade de ser económico e fácil de montar, possibilitando, por este motivo, a sua utilização em diferentes contextos, nomeadamente o ambiente de

casa ou de sala de aula.

Na impossibilidade de uma avaliação presencial e testar o sistema fisicamente, devido à pandemia do coronavírus (COVID-19), tornou-se necessário adaptar a fase de avaliação para que fosse possível obter algum tipo de resultados. Para isso, foram efetuados inquéritos com perguntas gerais e específicas do sistema, para serem respondidas com base na visualização do vídeo, que demonstra o funcionamento deste mesmo sistema, de modo a obter sugestões que possam ser mais benéficas para a sua melhoria e compreensão das vantagens e limitações (ver em anexo Questionário).

O *feedback* obtido foi muito positivo, uma vez que a maior parte dos inquiridos achou o sistema uma boa solução para o desenvolvimento computacional, podendo ser um grande auxílio para disciplinas como a matemática em que a lógica se encontra muito presente, e nas TIC (Tecnologia da Informação e Comunicação) em que são realizadas muitas atividades de robótica. Salienta-se que o sistema pode ser usado por crianças com um intervalo de idades muito alargado (entre os 7 e os 12 anos), mas que a componente relativa ao modo por voz, tem um público alvo mais jovem, sendo esta uma boa forma de os introduzir ao pensamento computacional e de desenvolver a lateralidade. Quanto às limitações do sistema, consideram que o ambiente de casa é mais favorável do que em sala de aula, devido ao facto de existir muito ruído, que pode dificultar o reconhecimento de voz, tendo sido sugerido o uso de *headphones*, pois poderia mitigar esta situação.

6.1 Trabalho Futuro

Nesta secção, é abordada a forma como se pode melhorar o sistema com base nos problemas e limitações identificados, bem como as sugestões fornecidas pelos inquiridos durante o processo de avaliação.

Uma das limitações está relacionada com a componente mapa físico e com o tamanho do *board*, pois os inquiridos consideram que este tem uma dimensão reduzida e que deveria ter uma dimensão maior, para que seja possível criar caminhos mais longos e complexos. A solução para esta limitação teria duas opções equacionáveis, ou se adicionam mais tabuleiros e colocavam-se os códigos *Topcode* [27] nos cantos, como referido anteriormente, ou imprimia-se em 3D um tabuleiro com uma dimensão maior, mas mantendo os códigos *Topcode* [27] nos seus cantos. Ao aumentar o tamanho implica um acréscimo do número de casas da matriz do *board* e, por esse motivo, seria necessário modificar na aplicação *Android* a quantidade de cortes na fotografia que a aplicação teria de efetuar, para a identificação da cor de cada casa e, desta forma, reconhecer o caminho.

Um dos problemas expostos relativamente ao reconhecimento de voz, foi o facto de apresentar uma velocidade lenta na obtenção das sequências, tendo os inquiridos sugerido que se realizasse de forma mais célere, pois as crianças podem perder algum foco e saturarem-se mais facilmente. Uma forma de mudar a limitação apontada, é a melho-

ria dos níveis, para que permita receber uma sequência de múltiplas instruções de uma vez só. De forma a melhorá-los será necessário efetuar uma análise dos inputs de voz na aplicação *Android*, para que seja possível contemplar variações no discurso, pois ao dar mais instruções de uma só vez, aumenta a complexidade no tratamento do discurso proveniente do utilizador. Para isso, deve ser atualizado no método *OnActivityResult()* da *activity MainActivity*. Depois, para agilizar a coordenação entre as questões da aplicação e o momento das respostas dadas pelas crianças, é preciso efetuar alterações na temporização da abertura da janela de reconhecimento, que se encontra programado no método *lauchSpeechRecognitionAccelaration()*.

Outra limitação apresentada foi a capacidade limitada, que o reconhecimento por voz pode ter no ambiente de sala de aula devido ao ruído que possa existir, não permitindo uma boa identificação do discurso da criança que está a usar o sistema. Neste caso, apontou-se que o sistema poderia incluir *headphones* para que se conseguisse ter uma melhor perceção auditiva das instruções do sistema. No entanto, não colmata a interferência causada no reconhecimento de voz, tendo para isso que ser utilizado principalmente como um auxiliar, em atividades de apoio, onde existem poucas crianças na sala. Outra alternativa seria o sistema ser requisitado por pais para ser utilizado em casa, uma vez que este se encontra preparado para essa eventualidade.

Para além das sugestões recebidas através dos inquéritos, considero que existem algumas questões a melhorar para que o sistema evolua. Uma dessas tem a ver com a precisão do reconhecimento da cor das diferentes casas da matriz, pois existem alguns fatores relacionados com a luminosidade que podem criar desvios, tal como sombras ou focos intensos de luz. Para solucionar esses problemas, devem ser criados algoritmos de correção da imagem usando a biblioteca do *OpenCv*, para que se possam identificar estes focos de luz e sombra, de modo a escurecer ou abrilhantar a imagem.

Considero que o sistema pode incluir mais elementos no jogo para dar à criança uma motivação extra, como recolher objetos para obter mais pontos e serem desbloqueados níveis, ao fim de um certo número de pontos.

O sistema poderia ter um ecrã junto ao *board* de *legos*, que possibilitasse o aparecimento da sequência de cores que a transmitisse ao robô, permitindo assim, uma melhor movimentação da criança sem ter que colocar o robô na superfície do *tablet*. Esta opção foi equacionada, mas as alternativas encontradas inviabilizaram a concretização de um sistema que pudesse integrar este tipo de ecrã, de forma a tornar o sistema menos dispendioso. No entanto, futuramente poderão surgir novas alternativas mais económicas.

Uma funcionalidade que penso que o sistema deveria ter, era os botões dos níveis conterem uma grelha com o caminho desenhado, para que fosse percetível a forma, como devem posicionar as peças no *board*, sem ter de recorrer a um guião para saber qual o caminho correspondente a cada nível. Para efetuar esta alteração, seria necessário modificar no ficheiro *XML* dos níveis, as *tags Button* para *ImageButton* e, posteriormente, definir o

caminho da imagem.

Considero interessante testar uma abordagem deste sistema num uso colaborativo, em que as crianças pudessem usar o sistema, competindo entre si, realizando, por exemplo corridas, tornando, assim, algumas atividades mais lúdicas e aprazíveis para as crianças.

Em suma, o *OzoUniverse* é um sistema que evidencia qualidades que possibilitam o desenvolvimento do pensamento computacional, em crianças cegas e com baixa visão. A aquisição destes conhecimentos é possível através de dois modos de interação inovadores como o modo por voz e o modo por blocos, que possibilitam a criação de sequências a executar pelo robô, que se movimenta sobre uma plataforma na qual é possível encaixar *legos* formando qualquer percurso favorável à deslocação do robô *Ozobot*.

Bibliografia

[1] Bitmap java object. https://developer.android.com/reference/android/graphics/Bitmap.

[2] Chaquopy framework. https://chaquo.com/chaquopy/.

[3] Dash. https://www.makewonder.com/.

[4] Doc. https://www.clementoni.com/pt/67285-doc-robo-educativo-falante/.

[5] Incode 2030 - educação. https://www.incode2030.gov.pt/atividades/educacao.

[6] Jotform platform. https://eu.jotform.com/.

[7] Lego mindstorms. https://www.lego.com/en-gb/themes/mindstorms.

[8] List object. https://docs.oracle.com/javase/8/docs/api/java/util/List.html.

[9] mbot. https://www.makeblock.com/mbot/.

[10] Ozoblockly. https://ozobot.com/create/ozoblockly.

[11] Ozobot. https://ozobot.com/.

[12] reactivision. http://reactivision.sourceforge.net/.

[13] String java object. https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html.

[14] Texttospeech object. https://developer.android.com/reference/android/speech/tts/TextToSpeech.

[15] Timer java object. https://docs.oracle.com/javase/7/docs/api/java/util/Timer.html.

[16] Matteo Abate, Lucia Pallonetto, and Carmen Palumbo. The effectiveness of motor activity on psychomotor development in school-aged children. 2020.

[17] Lúcia Abreu, Ana Cristina Pires, and Tiago Guerreiro. Tactopi: A playful approach to promote computational thinking for visually impaired children. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '20, New York, NY, USA, 2020. Association for Computing Machinery.

[18] Giulia Barbareschi, Enrico Costanza, and Catherine Holloway. Tip-toy: a tactile, open-source computational toolkit to support learning across visual abilities. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–14, 2020.

[19] Marina Umaschi Bers. *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge, 2020.

[20] Blockly. Blockly. `https://developers.google.com/blockly`.

[21] Gonçalo Cardoso. Ozouniverse - sistema que promove o desenvolvimento computacional em contexto educacional. `https://www.youtube.com/watch?v=M5aNmt9HOsg&t=10s&ab_channel=work`.

[22] National Research Council et al. *Report of a workshop on the pedagogical aspects of computational thinking*. National Academies Press, 2011.

[23] Georges Cuisenaire. *Les nombres en couleurs dans l'enseignement maternel et dans l'enseignement spécial*. Calozet, 1968.

[24] Pierre Debray-ritzen. *Dicionário de psicologia da criança*. Verbo, 1979.

[25] Fulano, Cicrano, and Beltrano. A paper on something. In *The 7th Conference on Things and Stuff (CTS 2009)*, Lisbon, Portugal, May 2009. Accepted for publication.

[26] google. Firebase. `https://firebase.google.com/`.

[27] Michael Horn. Tangible object placement codes. `http://users.eecs.northwestern.edu/~mhorn/topcodes/`.

[28] Michael S. Horn and Robert J. K. Jacob. Tangible programming in the classroom with tern. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '07, page 1965–1970, New York, NY, USA, 2007. Association for Computing Machinery.

[29] Michael S Horn and Robert JK Jacob. Designing tangible programming languages for classroom use. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 159–162. ACM, 2007.

[30] Felix Hu, Ariel Zekelman, Michael Horn, and Frances Judd. Strawbies: explorations in tangible programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*, pages 410–413. ACM, 2015.

[31] Hyunhoon Jung, Hee Jae Kim, Seongeun So, Jinjoong Kim, and Changhoon Oh. Turtletalk: an educational programming game for children with voice user interface. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2019.

[32] Kaarel94. Github. `https://github.com/Kaarel94/Ozobot-Python`.

[33] Varsha Koushik, Darren Guinness, and Shaun K Kane. Storyblocks: A tangible programming game to create accessible audio stories. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[34] Diego Lo, Paulo RS Mendonça, Andy Hopper, et al. Trip: A low-cost vision-based location system for ubiquitous computing. *Personal and Ubiquitous Computing*, 6(3):206–219, 2002.

[35] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):16, 2010.

[36] Sebastián Marichal, Anadrea Rosales, Fernando Gonzalez Perilli, Ana Cristina Pires, Ewelina Bakala, Gustavo Sansone, and Josep Blat. Ceta: Designing mixed-reality tangible interaction to enhance mathematical learning. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, New York, NY, USA, 2017. Association for Computing Machinery.

[37] Caitlin K Martin, Nichole Pinkard, Sheena Erete, and Jim Sandherr. Connections at the family level: Supporting parents and caring adults to engage youth in learning about computers and technology. In *Moving students of color from consumers to producers of technology*, pages 220–244. IGI Global, 2017.

[38] Oussama Metatla, Alison Oldfield, Taimur Ahmed, Antonis Vafeas, and Sunny Miglani. Voice user interfaces in schools: Co-designing for inclusion with visually-impaired and sighted pupils. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2019.

[39] Lauren R Milne and Richard E Ladner. Blocks4all: overcoming accessibility barriers to blocks programming for children with visual impairments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–10, 2018.

[40] Cecily Morrison, Nicolas Villar, Anja Thieme, Zahra Ashktorab, Eloise Taysom, Oscar Salandin, Daniel Cletheroe, Greg Saul, Alan F Blackwell, Darren Edge, et al. Torino: A tangible programming language inclusive of children with visual disabilities. *Human–Computer Interaction*, 35(3):191–239, 2020.

[41] OpenCV. Opencv library. `https://opencv.org/`.

[42] Osmo. Osmo. `https://www.playosmo.com/en/`.

[43] Ana Cristina Pires, Sebastian Marichal, Fernando Gonzalez-Perilli, Ewelina Bakala, Bruno Fleischer, Gustavo Sansone, and Tiago Guerreiro. A tangible math game for visually impaired children. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '19, page 670–672, New York, NY, USA, 2019. Association for Computing Machinery.

[44] Ana Cristina Pires, Filipa Rocha, Antonio José de Barros Neto, Hugo Simão, Hugo Nicolau, and Tiago Guerreiro. Exploring accessible programming with educators and visually impaired children. In *Proceedings of the Interaction Design and Children Conference*, pages 148–160, 2020.

[45] Alpay Sabuncuoglu. Tangible music programming blocks for visually impaired children. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 423–429, 2020.

[46] Wolfgang Slany. Catroid: A mobile visual programming system for children. In *Proceedings of the 11th International Conference on Interaction Design and Children*, IDC '12, page 300–303, New York, NY, USA, 2012. Association for Computing Machinery.

[47] Amanda Sullivan, Mollie Elkin, and Marina Umaschi Bers. Kibo robot demo: Engaging young children in programming and engineering. In *Proceedings of the 14th International Conference on Interaction Design and Children*, IDC '15, page 418–421, New York, NY, USA, 2015. Association for Computing Machinery.

[48] Anja Thieme, Cecily Morrison, Nicolas Villar, Martin Grayson, and Siân Lindley. Enabling collaboration in learning computer programing inclusive of children with vision impairments. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, DIS '17, page 739–752, New York, NY, USA, 2017. Association for Computing Machinery.

[49] Danli Wang, Cheng Zhang, and Hongan Wang. T-maze: a tangible programming tool for children. In *Proceedings of the 10th international conference on interaction design and children*, pages 127–135. ACM, 2011.

[50] Jeannette M Wing. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881):3717–3725, 2008.

[51] Junnan Yu, Clement Zheng, Mariana Aki Tamashiro, Christopher Gonzalez-millan, and Ricarose Roque. Codeattach: Engaging children in computational thinking through physical play activities. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '20, page 453–459, New York, NY, USA, 2020. Association for Computing Machinery.

Apêndice A

Questionário

Questionário sobre ambiente de programação tangível

Vimos convidá-la/o a participar num estudo de investigação acerca de um sistema para introdução à programação acessível a crianças com deficiência visual, programando um robot. O estudo consiste na visualização de um breve vídeo e no preenchimento de um questionário, que será apresentado de seguida, caso dê o seu consentimento.

Faço parte da unidade de investigação LASIGE da Faculdade de Ciências da Universidade de Lisboa e trabalho sobre a orientação do Prof. Doutor Tiago Guerreiro e da Doutora Ana Pires. Este estudo insere-se no âmbito da minha tese de mestrado que tem como objetivo desenvolver o pensamento computacional em crianças usando um ambiente de programação tangível e acessível.

Se tiver alguma preocupação sobre qualquer aspeto deste estudo, deve falar com o Prof. Tiago Guerreiro, que fará o seu melhor para o/a elucidar e responder às suas dúvidas, por e-mail, tjvg@ciencias.ulisboa.pt.

Se concordar em participar neste estudo, ser-lhe-á pedido que complete um questionário na página seguinte. Seguiremos todas as práticas éticas e legais e toda a informação sobre si será tratada de forma absolutamente confidencial.

Para prosseguir para a página seguinte do questionário, você indica que tem pelo menos 18 anos de idade, leu e compreendeu este formulário de consentimento e concorda em participar nesta investigação de forma voluntária e anónima.

Este questionário tem como objetivo recolher informação relativa ao sistema *OZOUNIVERSE*, demonstrado no vídeo abaixo. Este tem como público alvo crianças cegas ou com baixa visão e tem o objetivo de desenvolver o pensamento computacional, através da aprendizagem de conceitos de programação.

O *OZOUNIVERSE* contém uma plataforma que permite o encaixe de *legos* de texturas e cores diferentes, sendo possível a pais, educadores e às próprias crianças, criar qualquer tipo de mapa para que um robô designado de *Ozobot*, possa deslocar-se no trajeto delineado. Existem dois modos de programar o robô: um modo por blocos que utiliza peças tangíveis, e o modo por voz que através de um conjunto de questões é possível obter as

instruções a realizar.

Descrição do Sistema

Este questionário tem como objectivo recolher informação relativa ao sistema *OZOUNIVERSE*, demonstrado no vídeo abaixo. Este tem como público alvo crianças cegas ou com baixa visão e tem o objetivo de desenvolver o pensamento computacional, através da aprendizagem de conceitos de programação.

O *OZOUNIVERSE* contém uma plataforma que permite o encaixe de *legos* de texturas e cores diferentes, sendo possível a pais, educadores e às próprias crianças, criar qualquer tipo de mapa para que um robô designado de *Ozobot*, possa deslocar-se no trajeto delineado. Existem dois modos de programar o robô: um modo por blocos que utiliza peças tangíveis, e o modo por voz que através de um conjunto de questões é possível obter as instruções a realizar.

Caracterização Demográfica

1. Papel que desempenha
    - Pai / Mãe / Representante Legal
    - Familiar
    - Educador de necessidades especiais
    - Educador de TI / Robótica
    - Investigador (área de educação inclusiva) Outro
2. Trabalha ou já trabalhou com crianças com deficiências visuais?
    - Sim
    - Não
3. Tempo de serviço:
4. A(s) criança(s) são de que idade?
    - 1-3
    - 4-6
    - 7-9
    - 10-12
    - 13-14
    - Outros
5. Descreva-nos sumariamente a sua experiência como educador:

Perguntas Gerais do Sistema

1. Para que faixa(s) etária(s) considera que o sistema é adequado?
    - 1-3
    - 4-6
    - 7-9
    - 10-12
    - 13-14
    - Outros
2. Tendo em consideração as faixas etárias que escolheu na pergunta anterior, porque acha que são adequadas para a utilização do sistema?
3. Que qualidades mais apreciou na abordagem apresentada?
4. Que limitações ou problemas identifica na abordagem apresentada?
5. Em que disciplinas ou treino de capacidades pode ser aplicado este sistema?
6. Como imagina que este sistema pudesse ser utilizado em sala de aula?
7. Como imagina que este sistema pudesse ser utilizado em casa pelas crianças e eventualmente pelos seus encarregados de educação ou outros familiares (adultos ou crianças)?

Mapa do Sistema

1. Quais os benefícios e limitações que identifica nesta solução de criação de um mapa?
2. Que cenários/matérias/conceitos imagina poderem ser explorados em casa ou em ambiente de sala de aula com este tipo de mapa?
3. Tem alguma sugestão ou crítica relacionada com esta componente de solução?

Programação usando peças tangíveis

1. Quais os benefícios e limitações que identifica nesta solução de uso de tangíveis?
2. Como avalia a utilização de peças de lego para a criação dos blocos de programação?
3. Como avalia a utilização de peças de lego sobrepostas para a criação do conceito de ciclos (realizar repetições)?
4. Tem alguma sugestão relacionada com esta componente de solução?

Programação por voz

1. Quais os benefícios e limitações que identifica nesta solução da programação por voz?

2. Em que situações pensa que a programação por voz possa ser benéfica e para que faixas etárias?

3. Tem alguma sugestão relacionada com esta componente da solução?

Utilização do robô

1. Que impacto pode ter o tamanho do robô durante a utilização ?

2. Considera que deveria existir *feedback* para além do mostrado no vídeo ?

3. Tem mais alguma sugestão relacionada com esta componente da solução?